DEPARTMENT OF MECHANICAL ENGINEERING

QUEEN MARY COLLEGE

UNIVERSITY OF LONDON

DYNAMIC TESTING OF INTERNAL COMBUSTION ENGINES

by

EDWIN WILLIAM HISLOP

This thesis is submitted for the degree of Doctor of Philosophy in the University of London.

ABSTRACT

The automated testing of internal combustion engines is a field of considerable importance. This thesis describes a novel area in the form of dynamic testing. The investigation represents an original approach to the problem with which the author believes he has developed a new concept in engine testing.

The resulting information, combined with an appreciation of the latest developments in automation equipment technology, has lead to a review of the requirements of a generalised engine test system together with an outline of the way in which it could be implemented.

The automated engine test beds used during the work are described. The conventional methods of testing internal combustion engines are reviewed and a generalised approach to automating them proposed. This then leads to the suggestion of a new method using dynamic testing techniques specially evolved in the course of the work. On the basis of this the implementation of a dynamic full-load power curve as a replacement for its steady state equivalent was pursued.

The second major use of dynamic techniques is for exhaust emission cycles. Both dynamic (USEPA Smoke Cycle) and steady-state (USEPA 13 Mode Cycle) cycles were performed. In the case of the former, outer digital loop techniques were used to improve control response.

In performing the above work, use was made of both analog and digital simulation techniques for development purposes. This work was also extended by the development of a simulation of a material handling system to enable the control and optimisation of a production test

facility to be studied. The testing methods associated with automated fault diagnosis are also analysed.

On the basis of the above work an engine test system task analysis was generated and this new concept used to plan a microprocessor based engine test automation scheme.

To my parents,

For their continuous help and

encouragement throughout my education

ACKNOWLEDGEMENT

I should like to thank Dr. J.I. Soliman, my supervisor, for giving me the opportunity to perform this research project and for all his guidance and encouragement during its course.

Also I am grateful to all those persons at Queen Mary College who have helped in one way or another over the years and in particular to Mr. W. Szatkowski, Chief Experimental Officer.

I should also like to acknowledge the contribution of the Ford Motor Company in allowing me to use their facilities and to thank the members of their staff, in particular Mr. I. Farrow and Mr. A. Boyle, for their cooperation.

CONTENTS

CONTENTS (cont.)

CONTENTS (Cont.)

Chapter 1

INTRODUCTION TO DYNAMIC TESTING

During the past few years we have seen a tremendous breakthrough in the whole field of industrial automation. The prime element in bringing about this change has been the revolution that has occured in the manner in which our industrial test and control equipment is fabricated. The advent of digital integrated circuits has meant that smaller and cheaper computers have become available allowing automation schemes to be put forward for many applications that would not previously have been thought viable. There have in fact been two complementary factors at work here, for whilst we have seen the decline in the cost of our automation equipment, at the same time there has been a steady increase in the cost of labour, particularly of the skilled or semi-skilled variety. This has meant that the trend over the past decade has been to consider ways in which machines can replace human labour. Even where it is not possible to replace the man entirely, either for technical, social, or political reasons, improved, highly automated, equipment can allow us to increase the productivity of our work force whilst at the same time bringing about improved working conditions and greater margins of personal safety.

The use of automation in the field of engine testing has been growing slowly for some considerable time. Some of the earliest attempts owe little or nothing to modern integrated circuit based technology. With the coming of the minicomputer however, work on the production of high level automation schemes really got underway. At first there were many development problems and it was not until the early and mid-seventies that we began to see systems actually installed and working in industry despite the fact that minicomputers

had already by then been available for quite some time.

During the course of this development period, other factors were coming to light which were to have a profound effect on the importance of automated engine testing. The early seventies saw an increased consciousness world-wide of the harmful effect of exhaust emissions on the environment. As a result there has been a tremendous amount of legislation requiring manufacturers in many countries to attain strict limits as far as emissions are concerned. Another facet of life in recent years has been the increased importance attached by all concerned to the economy of fuel consumption of our motor vehicles as a result of problems in the world-wide availability and cost of oil-based products.

Traditionally our procedures for testing internal combustion engines have inevitably been heavily influenced by the actual equipment available. In the past to measure the performance of an engine it would be necessary for the operator to go round examining various gauges and dials and writing down the readings. Because of the time taken to do this for all the necessary data parameters, the basic test procedures were all of a steady-state nature. The engine would be brought to a specific operation condition (speed, throttle position, etc) and allowed to settle. Only after a complete set of readings had been logged manually could the test proceed to the next stage.

Early work on automated engine testing was largely based on making the computer take over the duties of the human operator running the traditional tests. The advantages of this were two-fold.

Firstly it was hoped that there could be an overall saving in man-power with the computer operating several test beds simultaneously, thus reducing the number of operators required below the one per bed needed for manual operations. The second advantage was that the high speed data collection abilities of the computer could shorten the testing time and hence increase the amount of work a single test bed could undertake.

It is this latter point that is the first indication of the new horizons opened up by our use of automation. The old limitations on information gathering and manual control do not apply in a computer based automation environment. As a result it has become possible to re-think completely the procedures used in engine testing. We are no longer constrained to the use of steady state conditions and so we can introduce the concept of dynamic testing. With the development of highly variable control abilities and practically instantaneous data logging, engine conditions can be varied rapidly with respect to time whilst allowing virtually any amount of data retrieval to be performed.

There have been two basic factors which have given rise to the importance of dynamic testing methods. The efficiency with which a test bed can be used has always been important. Even in the case of the earlier automation schemes which were almost entirely for research and development purposes, this was the case. However one of the developments recently has been the use of automation in production testing facilities and here maximised throughput is at a premium. As is shown later on the replacement of traditional steady-

state tests by their equivalent dynamic versions provides a technique to reduce the time duration of a given test and hence increase our overall efficiency.

The second factor at work here has already been mentioned as it is the increasing importance of emissions and fuel economy testing. The static conditions of steady-state testing do not represent truthfully the typical usage that an engine gets in real life. Instead, most driving, particularly when of an urban nature, is composed mainly of transients of speed and load. Furthermore it is these transients that are often the areas of worst performance as far as emissions and fuel consumption are concerned. As a result of this, dynamic testing of the type that allows normal road usage to be simulated, has become vital. As far as emission testing as a result of legislation is concerned, the development of highly sophisticated dynamically orientated automation systems has become vital if the accuracy needed to give a reliable and valid comparison between different engines is to be met.

The research work described in this thesis covers a span in time from the infancy of minicomputer-based automation of engine test beds up to the present day. After seeing the actual installation of the first generation of systems in industry, a considerable amount of work was performed in investigating the use of dynamic testing techniques with a view to improving the performance of the second generation of systems. At the same time consideration was given to various concepts that had not been included in the first generation such as engine malfunction diagnosis.

While this work was going on, new problems were met. The two major items here were that results were beginning to come in from the experience of actual industrial usage of first generation systems and also that digital processing equipment was undergoing a revolution with the advent of the microprocessor. The result of all this was that it became necessary to re-think our system philosophy. Operating experience had shown that the overall software and hardware architechtures used in the first generation had considerable drawbacks and this combined with the abilities needed to perform dynamic testing has lead to the planning of a completely new philosophy for second generation systems.

The work described in this thesis together with the accompaning results and conclusions drawn, is seen as providing the bridge between the first and second generations of engine test automation systems, by providing much of the necessary ground work.

Chapter 2

SURVEY OF PREVIOUS WORK

2.1 Introduction

The automated testing of internal combustion engines is a field that has seen the publication of a considerable volume of literature. Despite this, the number of published papers directly relating to the subject of this thesis is comparatively small, no doubt due to the fact that work on dynamic testing is a new field and that the investigation described represents an original approach. Thus in the following survey of previous engine testing, most of the papers concern the earlier work on conventional testing systems and serve to provide a background against which the development of dynamic testing can be compared.

2.2 Automated Engine Testing

1) Mobil Oil Company - Two papers published in 1972 by Gardiner, Heckford and Lowres (20) and by Gardiner and Price (21) describe the engine test facility orginally built. The type of testing performed is steady-state in nature with slow response times required during state changes. Thus the system of using a single computer to perform direct digital control of 15 test beds is only viable in this type of control but does not give us much assistance in implementing dynamic testing systems. A later paper (22) by Gardiner, Southgate and Rea presents the subsequent re-computerisation of this facility. As the basic testing requirements remained unchanged the new system is similar in concept to its predecessor and has the same drawback. An interesting factor however is the way in which

a standardised set of process control software has been used to implement the system. This probably represents the ideal sort of application for the standard package approach.

2) General Motors - The automated performance and durability test system of Detroit Diesel Allison is described in a paper by Henderson (32). This system is primarily used for data logging and reduction but does have automated control abilities. The paper states that it was planned to use the system to perform emission test cycles including the USEPA Smoke Cycle but as the system has a single processor for 22 test beds and only updates set points to the local controllers at 100 millisecond intervals it is difficult to see how this could be accomplished.

3) General Automation - A paper by Kibble (37) provides a proposal for a multi-cell exhaust emission test system capable of performing dynamic tests. The general concept of the system organisation is well thought out but is based on a single minicomputer providing speed set points for local controllers. There does not appear to have been any consideration given to the actual control problem involved and the use of outer digital loops is not mentioned. As will be seen later on this would involve unacceptable requirements for the performance of the local control units.

4) Siemens - It is worth mentioning two papers on test rig automation from Siemens. The first, by Aggett and Heck (5), advocates the use of a standardized software package. It is also interesting for its discussion on organisation of hierarchical

systems controlling many test beds, although, using minicomputers, it only goes as far as having a minimum of two test beds per processor. In such an environment, a certain amount of dynamic test work could be accomplished but complex or difficult dynamic tests would still be precluded.

A subsequent paper by Bongiorno and Bischof (16) describes the Siemens standard package system in more detail and argues its advantages. Of particular interest is the estimation of the way in which the system costs divide up between software and hardware and in increasing percentage of the software part with time. The paper also states that using the standard package approach can reduce programming time by up to 30%.

5) International Harvester Company - A paper by Grunert, Schultz and Ackermann (30) describes this comparatively new facility. Although some mention is made of emission measurements, it is presumed this only applies to steady-state testing. The system has eight test cells under the direct digital control of a single minicomputer and it is difficult to see how execution time requirements could also allow anything other than a fairly slow response time.

6) GEC Elliot Process Automation - One of the best examples of the use of a standard software package can be seen in the paper by Lowres (39) which describes the CONRAD package. An important feature is the description of an actual facility using this system. Nevertheless, the basic problem with all such systems still exists, namely, the difficulty in providing the very fast response times

required for dynamic testing. The problem of precision sequencing of dynamic set points exists in such a system as much as it does for a specially programmed one and in addition the response is worsened by the need for considerable background overheads in processing time.

7) Volkswagen - The major engine test system of Volkswagen is described in papers by Bender (13) and Schulz and Mund (47). The 76 test-beds are controlled by local analog controllers and six minicomputers each handling a group of test cells, which in turn communicate with a large supervisory computer. Due to the connection of several test-beds to each minicomputer the system is essentially intended for steady-state testing. However the application of this philosophy to dynamic testing can be seen in a paper by Schweimer (48) describing a system to perform road drive simulation and emission cycles. The system uses a single processor to output set points to local analog control systems. Although no mention is made of any outer digital loop control so that it is hard to see how any shortcomings of the local controllers are dealt with, the use of steady state recalibration of the measuring systems immediately prior to running a dynamic test is an interesting concept.

8) Obr Wsk Mielec and Politechnika Gdanska - One of the earliest examples of the application of the microprocessor to automated engine testing is the paper by Bialynicki, Cichy and Mazurek (14). The basic concept is very advanced and uses the idea of a single low-cost processor for each test-bed. Regretfully though, the system performance has been limited to steady-state testing both by software

and hardware considerations. Particularly the use of a magnetic cassette tape system to generate set points represents something of a step backwards rather than forwards.

9) SEPA - A more recent description of the use of a microprocessor system can be found in a paper by Bonamico (16). This system again uses a single microprocessor per test-bed. The actual control is performed by local analog controllers whilst functions such as data storage and processing are performed by a supervisory computer. It appears that at present the system is limited to steady-state testing but with the provision of adequate software there is no reason why dynamic testing could not be performed. It is also suggested that a larger proportion of the supervisory computer functions could be implemented in the microprocessor instead thus reducing its real time response requirements and increasing the number of test-beds connected to each computer beyond the five to ten quoted.

10) General - Examination of the literature sources the above examples have been taken from will reveal many more papers on engine testing automation. In the main there is little point in discussing all of them here as they merely describe the systems used by different organisations and are very similar in overall terms. It is probably true to say that most of them concern systems where a single processor is used in a multi-cell environment and hence, whilst being capable of steady state testing, they are not really viable systems for dynamic testing.

The reader is also referred to the publications on earlier work

at Queen Mary College (6, 9, 28, 29, 54, 55, 56, 59, 60).

2.3 Dynamic Testing

Papers specifically concerned with the problems of dynamic testing, other than the authors' own publications (34, 35, 50, 51, 52, 53), are somewhat rare but some references are given below.

1) Ford DRIVER - The Dynamometer Road Imitator and Vehicle Economy Rater (D.R.I.V.E.R.) is a system designed for use in dynamic tests for fuel economy and exhaust emissions and is described in a paper by Farrow (18). The system using a D.C. electric dynamometer and associated Ward-Leonard set to brake or motor the engine. Local control circuits derive their set points from a punched tape reader. As the maximum rate of set-point updates is once a second, considerable reliance must be placed on the operation of the local controllers. At present the system is not capable of performing the more complex emission cycles with sufficient accuracy. The analysis carried out by Wahba (60) has shown that theoretically the system can meet the necessary accuracy demands but this has yet to be proven in practice. Furthermore the setting up of optimum controller gains requires complex modelling and analysis of the system (including the engine) and hence limits the flexibility of the concept.

2) Lucas - One of the major problems in dynamic control is the interaction between different engine variables. The paper by Ironside (36) provides an important insight into how a control system for dynamic work can be designed. The use of analog techniques in

the controller seems somewhat over complicated but there is no reason why the function could not instead be implemented in a digital algorithm. In addition, as the design of the controller was based on the mathematical analysis of a particular engine and test bed combination, it will be important to see how well the technique performs from a point of view of general applicability.

3) Istituto Motori - A method of using simulation techniques to design optimal control strategies is put forward in a paper by Gabola, Migliaccio and Innocente (19). The basic concept is that of using a very flexible strategy allowing a large number of different control loops to be used in conjunction with three-term analog controllers. The paper does not give any real indication of how such an approach performs in practice and it is felt that problems may be encountered due to the simplification and idealization of the functions of some of the models elements, in particular the controllers themselves.

2.4 Diagnostic Testing

1) Hamilton Standard - As one of the leading manufacturers of automated engine test systems and specialists in engine diagnosis, various papers from this organisation provide an interesting insight into the various techniques of engine fault diagnosis. The paper by Goodfriend (27) provides specific examples of diagnostic methods, particularly those using high speed data logging to measure the variation of a particular parameter with crankshaft angle. The only

disadvantage of the system described is the large number of transducers used, for example to provide intake and exhaust valve timing and bearing malfunction testing, a total of 13 sensors points (accelerometers) are used. It is felt that more comprehensive mathematical analysis from fewer, more generalized sensor points could also provide the same information.

A later paper by Prue (44) provides a wider ranging description of diagnostic testing methods. Besides the previously mentioned variation with crankshaft angle methods there are also a number of tests involving the measurement of how a parameter changes over the engine speed range and this would seem to be an ideal application for dynamic testing along the lines proposed in Chapter 5.

2) Ono Sokki - A paper by Maeda, Ono and Ohigashi (40) describes the way in which Fourier analysis can be used to diagnose engine faults. The actual parameter analysed is noise but there does not appear to be any reason why the technique could not also be used in conjunction with other signals. It would be interesting to hear if any automated, as opposed to manual, method has been developed for the extraction of the relevant features from the various frequency power spectra.

3) Ship Research Institute of Norway - A somewhat different approach to diagnosis is characterised in a paper by Sandtorv, Fiskea and Gundersen (46). The technique here is specific and detailed instrumentation of various engine components. It is felt that this technique does not hold any great promise owing to the difficulty of

fitting such instrumentation, particularly in the case of smaller engines than the marine diesels referred to in this paper.

2.5 Modelling and Simulation

There are many different approaches to modelling of internal combustion engines. In connection with this investigation we are mainly concerned with the overall simulation of the engine and dynamometer system in real time so that we can ignore detailed studies of internal process modelling such as combustion analysis.

1) Queen Mary College - A considerable amount of work on real-time simulation of engines has already been performed. Two somewhat different approaches to the problem can be seen in the following examples. The paper by Gravestock (28) shows how a simulation can be built up from the examination of the internal processes of the engine. This provides a very realistic model even to the point of allowing the simulation to include the effects of warm-up on a cold engine. A possible criticism is that the resulting simulation is unnecessarily complicated and does not include any modelling of the other components of the test bed such as the throttle actuator and dynamometer.

An alternative approach used by Al-Bermani (6) is to regard the test bed as a few 'lumped' systems and to analyse their transfer functions on the basis of practical experiments on a test bed. The findings can then be used to construct a simulation model. This technique has the obvious advantage that simulation results can be correlated easily with actual engine behaviour but it must be remem-

bered that the validity of the simulation may be limited to the conditions under which the transfer function was obtained. For example in this case the simulation model does not make allowance for variation in engine and ambient temperatures.

2) University of Naples - Whilst generalised models of engine/test bed systems as described above provide a useful tool in general development of automated test systems, they do not also correlate well in dynamic situations. To improve their performance in this area, it might be possible to include in the simulation a more detailed model such as that put forward by Balestrino, Eisinberg and Sciavicco (10). The problem with such modifications is that the complexity of the model increasingly requires more extensive facilities to allow real time simulation.

2.6 Handling Systems.

As the detailed and specific simulation of a handling system described in Chapter 8 is a special, and as far as the author is aware, original piece of work, there would not appear to be any particular merit in surveying more generalized simulation methods that are sometimes involved in this area. Similarly, the majority of conventional handling system techniques such as overhead conveyors, drag-lines, etc. are too well known to be worth describing in detail. Instead it is proposed to present examples of developments in handling system technology that may well represent the basis for future highly automated engine test facilities.

1) SI Handling Systems - The paper by Spooner (58) describes what is basically a development of the pallet and conveyor system. The advantage of the system is its inherent positional accuracy, a factor that is vital if such techniques as automatic coupling of an engine to the test bed are to be used. On the other hand the system requires a large amount of complex, fixed track which may provide problems for the point of view of cost and accessibility.

2) Digitron - A contrasting approach which does not use any track at all is put forward by Mueller (41). This system uses robot trailers following imbedded guide wires in the floor. This obviously has a reduced track cost but will result in more expensive pallets and may need additional locating equipment to obtain the necessary positional accuracy when being linked to a test bed.

2.7 Summary

The various publications included in this survey have been selected in order to set the scene for the following chapters. The most apparent fact is that the field of dynamic testing is still very much in its infancy and although there are some papers that discuss the problem, there is a lack of actual results and proven examples of the use of dynamic techniques. The majority of engine test systems are based on the concept of a single processor handling the responsibility of several test beds and as will be shown later on this does not form an adequate automation structure for most dynamic testing.

Chapter 3

AUTOMATED ENGINE TEST EQUIPMENT

3.1 Introduction

Although the work described in this thesis covers many different aspects of engine test automation, most of the experimental part of the work was performed at one or other of two facilities. The first was at Queen Mary College, whilst the second, based largely on the college's design philosophy was situated at Ford Motor Company's Dunton Research and Engineering Centre. A description of both types of test stand with the associated background working that went into their planning is included in this chapter. Because of the flexibility of a computer based system the same basic experimental hardware could be used for many different types of experimental work with only, at the most, minor modifications to the hardware. For this reason the other chapters describing the experimental work concentrate on the software used and the results obtained and let the reader refer to this chapter to obtain information on the equipment.

3.2 Internal Combustion Engine Control

In general we may consider our whole test rig as a single unit to which we feed various control signals and which in return provides us with various outputs reflecting what is happening. To simplify matters we can divide our parameters into two groups, primary and secondary. The four primary parameters are those we shall be most concerned with in controlling and monitoring our test bed. Under the heading of secondary parameters come all the various temperature, pressure and other miscellaneous variables involved in engine testing.

The four primary parameters consist of two inputs, throttle position and dynamometer loading, and two outputs, engine speed and torque. The whole system can be lumped together as a single unit (see Figure 3.1) with the variables ϕ, L, N and Q representing the four parameters.

In a majority of test applications our prime task is going to be to monitor and control these parameters independently from any work associated with the secondary parameters. There are basically 3 different ways in which we combine the inputs and outputs to give us our modes of control as shown in Figure 3.2. These are:-

i) Mode I - The speed is controlled by alteration of the dynamometer loading. At the same time the torque developed is set by control of the engine throttle position.

ii) Mode II - In this mode the inputs are reversed so that the throttle position is used to control the speed and the dynamometer loading controls the torque.

iii) Mode III - Here the speed control loop is identical to that of Mode I, i.e. speed controlled by alteration of the dynamometer loading, but the throttle position is set directly and the torque thus becomes a dependent variable.

Obviously for a great many tests such as full and part load power curves and the USEPA Smoke Cycle which require definite throttle settings and speeds with the resultant torque to be measured, Mode III is the most convenient. In cases where the only throttle settings involved are fully opened and fully closed it is, of course, possible to

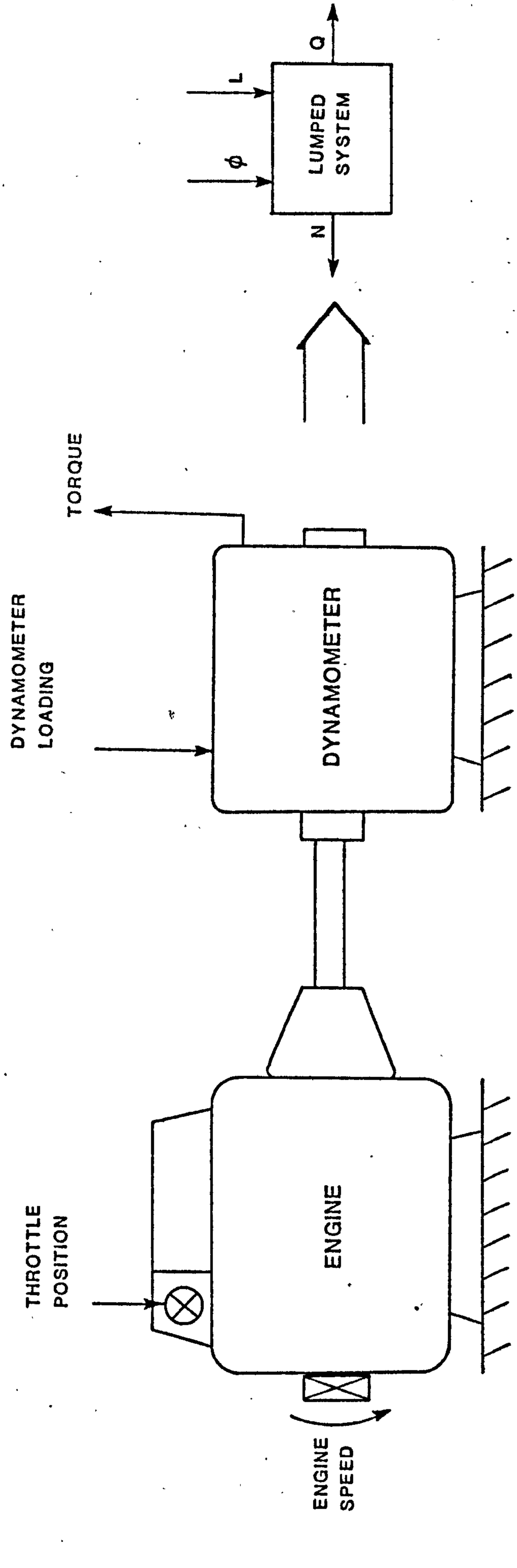


FIGURE 3.1 - ENGINE CONTROL PRIMARY PARAMETERS

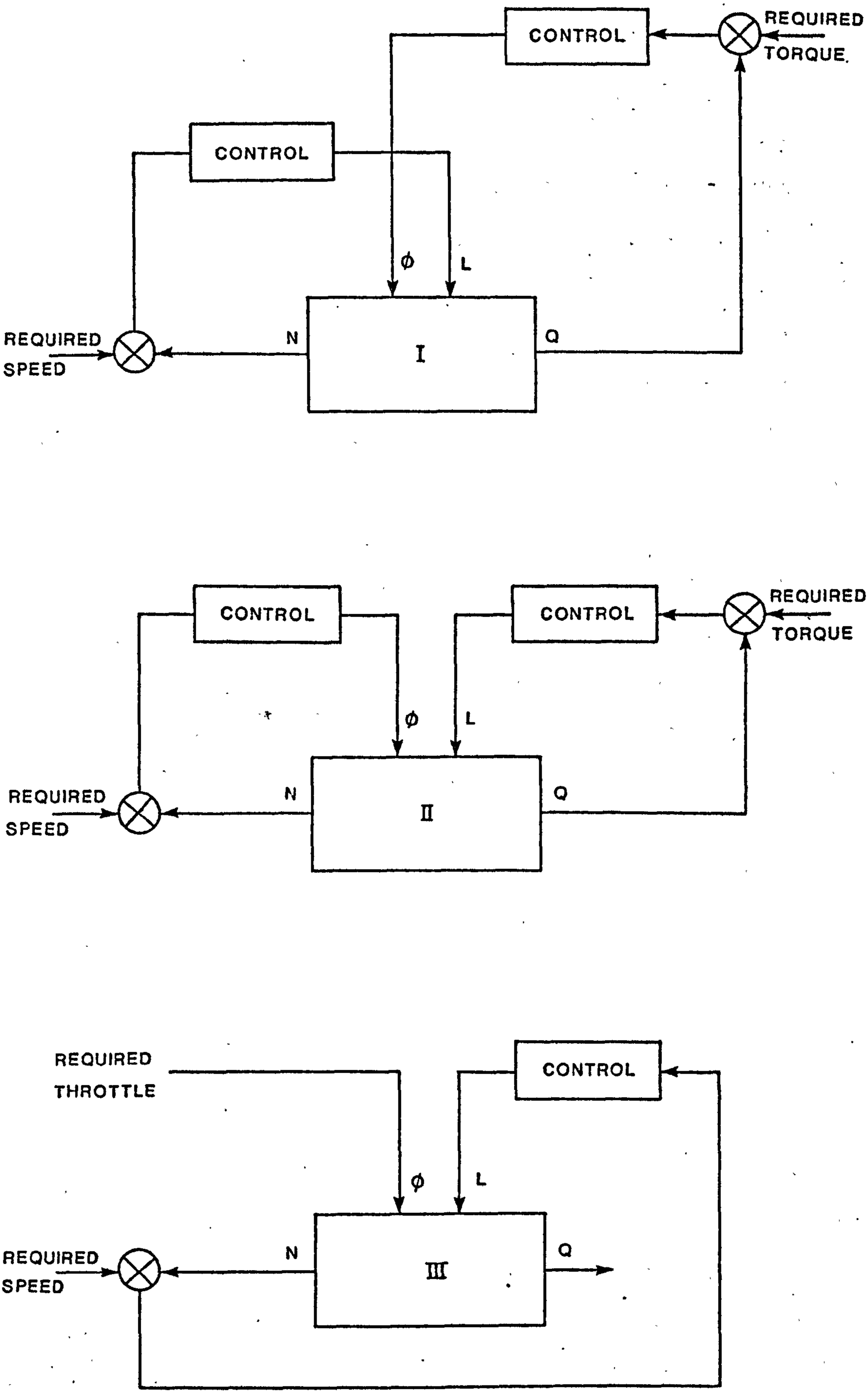


FIGURE 3.2 - MODES OF CONTROL

use Mode I by specifying zero torque required for fully closed throttle and a greater torque than the engine can possibly produce for fully open throttle.

Test procedures that define both the torque and speed set points can use either Mode I or Mode II. In practice one or the other will prove the better depending on the relative speeds and stability of the two control loops in relation to the changes in set points required by the test procedure.

The actual physical realisation of these control modes can be based on two alternative techniques, direct digital control (DDC) within the computer itself or alternatively using local analog controllers with the computer providing only the set point. In both of the facilities described here the latter approach was used as this allowed the computer software to be simplified and hence meant that the computer was capable of using multi-programming to control several test rigs simultaneously if the need arose.

3:3 Queen Mary College Test Beds.

The overall structure of the test beds set up at the College is shown in Figure 3.3. The computer and associated equipment was housed in a separate computer room immediately above the laboratory containing the actual test rigs (see Figure 3.4.).

The computer used was a General Automation SPC-16 minicomputer (26) with 16K words of core memory. The system input/output was based on two ASR33 teletypes and a high speed paper tape reader and punch. In single test rig work, teletype 1 was used as the overall

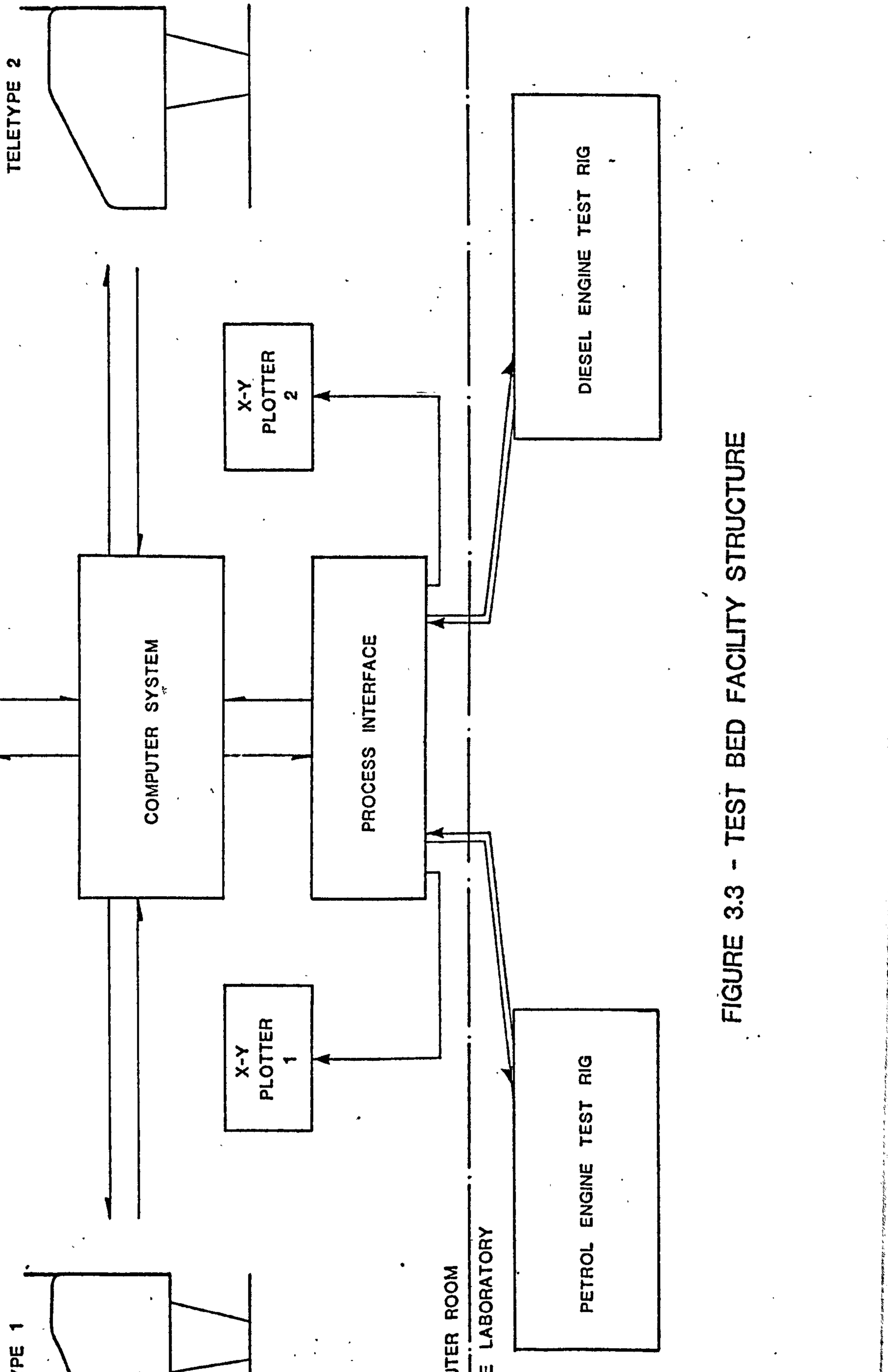


FIGURE 3.3 - TEST BED FACILITY STRUCTURE

computer/operator interface, whilst in multiple test work it handled the petrol engine test rig, with the second teletype being used for interface with the diesel engine test rig.

The computer system also included a process interface containing digital - to - analog (DAC) and analog - to - digital (ADC) conversion as well as direct digital inputs or outputs (DDI and DDO). Most of these channels were used to interface the computer to the test rigs but some of the DAC channels are used to drive two X - Y plotters, one for each test rig. The monitor of a closed circuit television system enables the operators to use a remote controlled camera in the engine laboratory to observe the behaviour of the test rigs. Figure 3.4 also shows the second computer system based on an EAL 640 computer that was used for tape preparation, simulation and other off-line type activities.

The basic schematic of the test rig design which was used, is shown in Figure 3.5. The diagram, together with the description below, applies to both test rigs unless otherwise stated. The engine speed was monitored by a magnetic sensor and toothed wheel combination. The resulting pulse train was converted to an analog voltage signal which could be monitored by the computer's ADC interface. This signal was also fed to the signal conditioning and switching array. The primary control loops were based around two Motorola 3-term analog controllers. The output of one was fed directly to the drive controller of the Heenan and Froude Dynamatic Mk I eddy-current dynamometer. The engine's throttle position was operated by a specially designed throttle actuator based on a

FIGURE 3.4 - COMPUTER ROOM


EAI 640
NO SMOKING

FIGURE 3.4 - COMPUTER ROOM

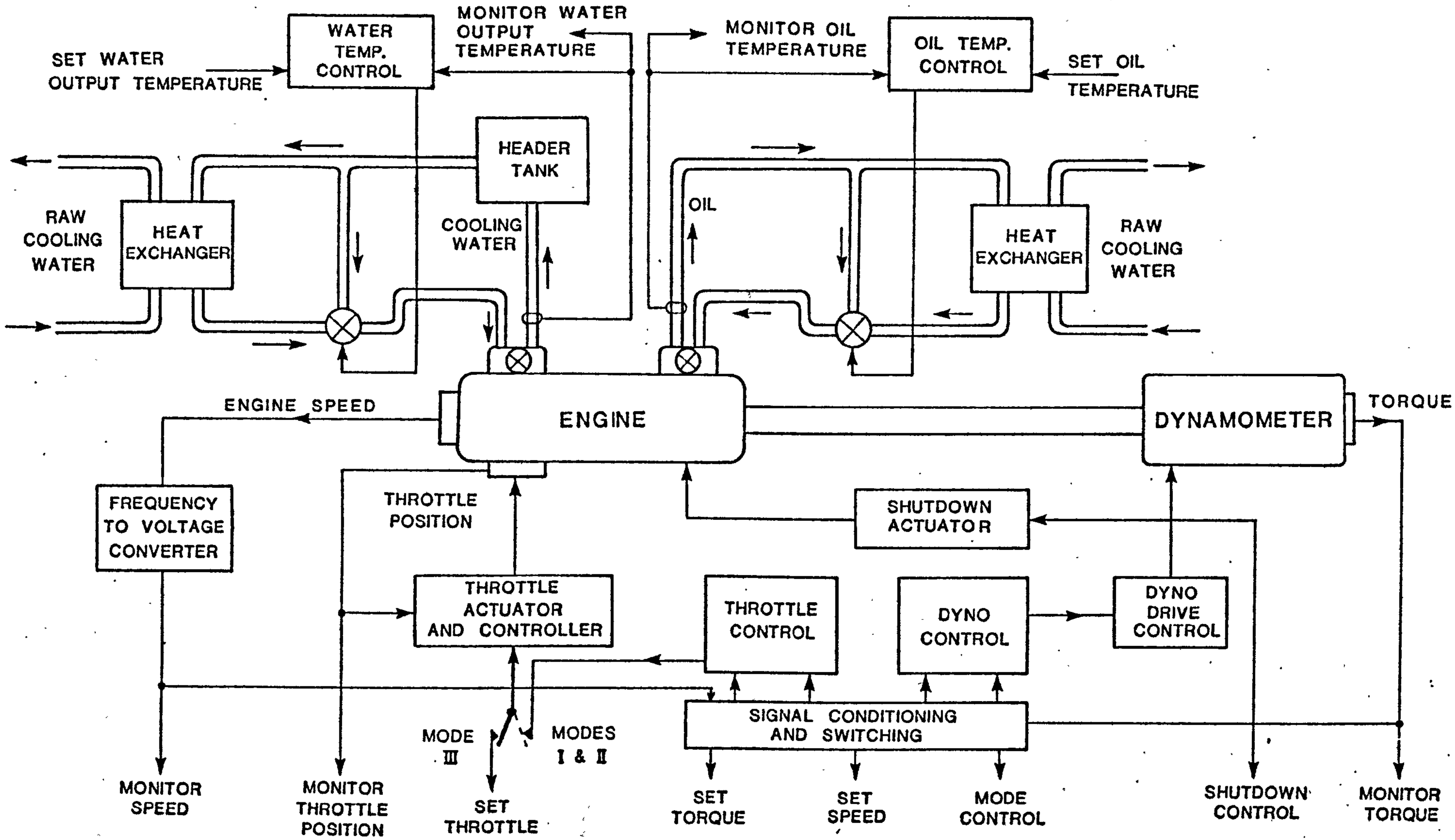


FIGURE 3.5 - TEST RIG MONITORING AND CONTROL

Printed Circuit type electric motor (31). Both test rigs used this type of actuator but a larger and more powerful motor was used in conjunction with the diesel engine to provide the high torque required to operate the injector pump speed lever combined with the fast response necessary for some dynamic testing in connection with exhaust emission cycles. The associated controllers were also specially designed and built.

The throttle position set point had two alternative sources. It could be set directly by an output from one of the computer DAC channels or by the output of the second local analog controller. The actual source was under the control of a relay operated by the computer digital outputs.

The two rigs used slightly different methods of sensing the developed torque. In both cases an indirect method, measurement of dynamometer carcase displacement, was used. The petrol engine test rig used a load cell connected between the dynamometer carcase arm and its bed-plate. For the diesel test rig, the load cell was replaced by a proving ring and a linear variable differential transformer (LVDT).

Both types of torque measuring system produced an analog voltage proportional to the developed torque (see Chapter 5). As well as being made accessable to the computer system via one of its ADC channels, this signal was also fed to the signal conditioning and switching array. This array consisted of a number of analog circuits to perform level shifting and general signal conditioning. In addition, several relay based switching circuits could be used for the mode

control from the computers DDO's to generate the required set points and process connections to the controllers resulting in the desired mode of control.

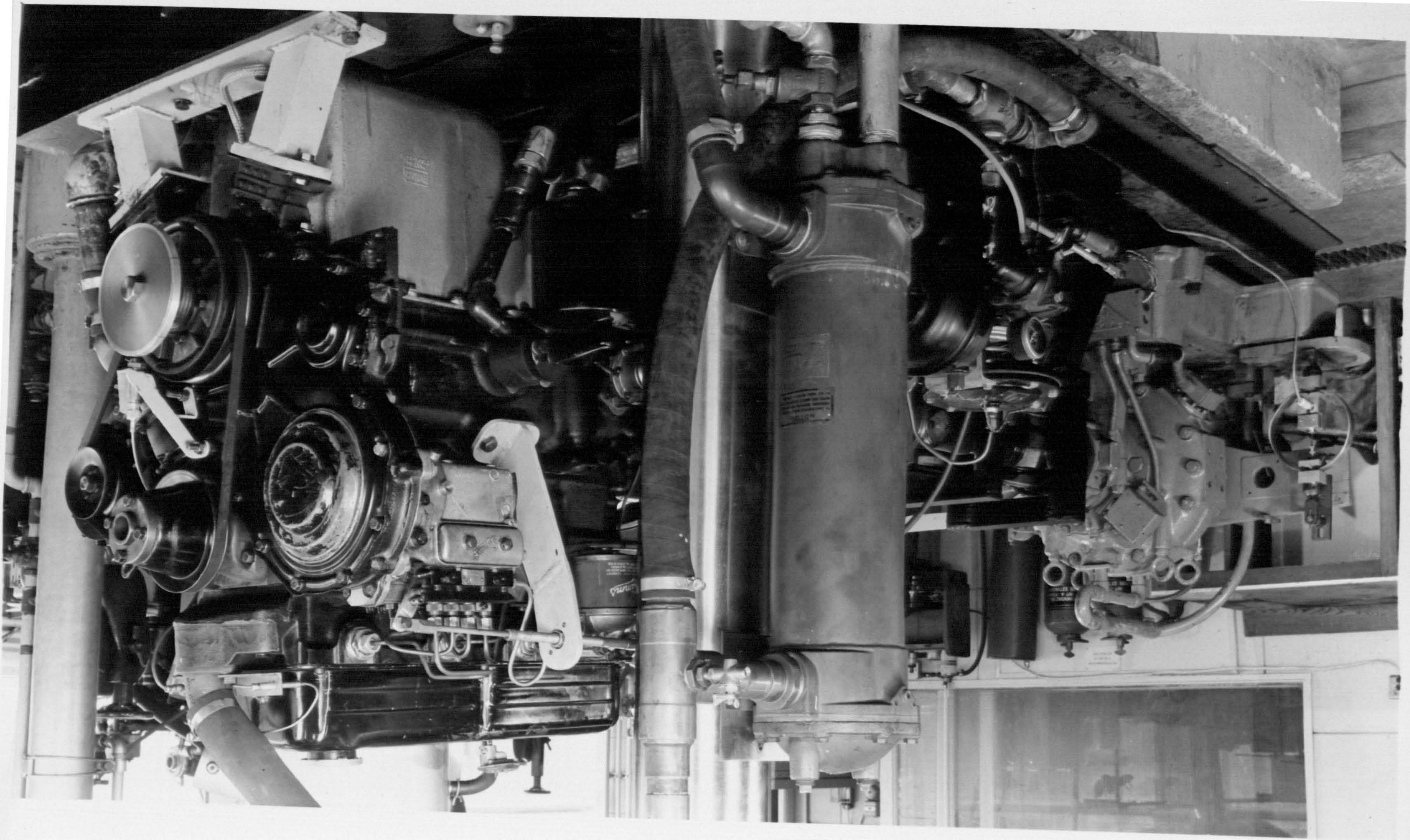
As a safety measure starting up of the engines could only be accomplished by manual action of a starter button on the test rig. Both test rigs had a form of automatic shutdown system. In the case of the petrol engine rig this was simply a device that disabled the ignition circuit. The diesel engine rig, on the other hand, was provided with a hydraulic based system to operate the shutdown lever on the injector pump. As can be seen from Figure 3.5, the shutdown system could be operated by the computer using a DDO channel. In addition the circuit was designed to fail-safe (i.e. engine stopped) under a number of other conditions, such as low pressure in the cooling water supply or lack of exhaust extraction, as a result of hardware sensors.

As well as the primary test rig parameters, it was also necessary to control the engines oil and cooling water temperatures. Figure 3.5 shows the basic set-up used to accomplish this. Once again the heart of the control loop was a Motorola 3-term analog controller (thus a total of 4 were required for each test rig). The temperatures were sensed using Chromel-Alumel junction thermocouples with associated amplifiers and an ice-point reference. The control action resulting from comparison of the temperature with a set point generated either locally, or by the computer, operated a mixing valve. By varying the amount of flow of liquid returning directly to the engine with that diverted through a heat exchanger, the temperature could be controlled.

PARAMETER	TEST RIG	SENSOR	OUTPUT
SPEED	BOTH	ELECTRO-MAGNETIC AND F-TO-V CONVERTER	ANALOG
TORQUE	PETROL	LOAD CELL	ANALOG
TORQUE	DIESEL	PROVING RING AND LVDT	ANALOG
THROTTLE POSITION	BOTH	POSITION POTENTIONMETER	ANALOG
TEMPERATURE i) WATER INLET ii) WATER OUTLET iii) OIL iv) AIR INLET v) EXHAUST	BOTH	CHROMEL-ALUMEL JUNCTION THERMOCOUPLES, LOW DRIFT D.C. AMPLIFIERS AND ZEREF ICE POINT REFERENCE	ANALOG
PRESSURE i) OIL ii) EXHAUST	BOTH	STRAIN GAUGE PRESSURE TRANSDUCERS AND BRIDGE	ANALOG
SMOKE	DIESEL	AVL 412 FILTER TYPE SMOKE METER	ANALOG
FUEL CONSUMPTION	PETROL	AVL SYSTEM 700 FUEL CONSUMPTION MEASURING EQUIPMENT	DIGITAL

FIGURE 3.6 - TEST RIG DATA PARAMETERS

FIGURE 3.7 - DIESEL ENGINE TEST RIG



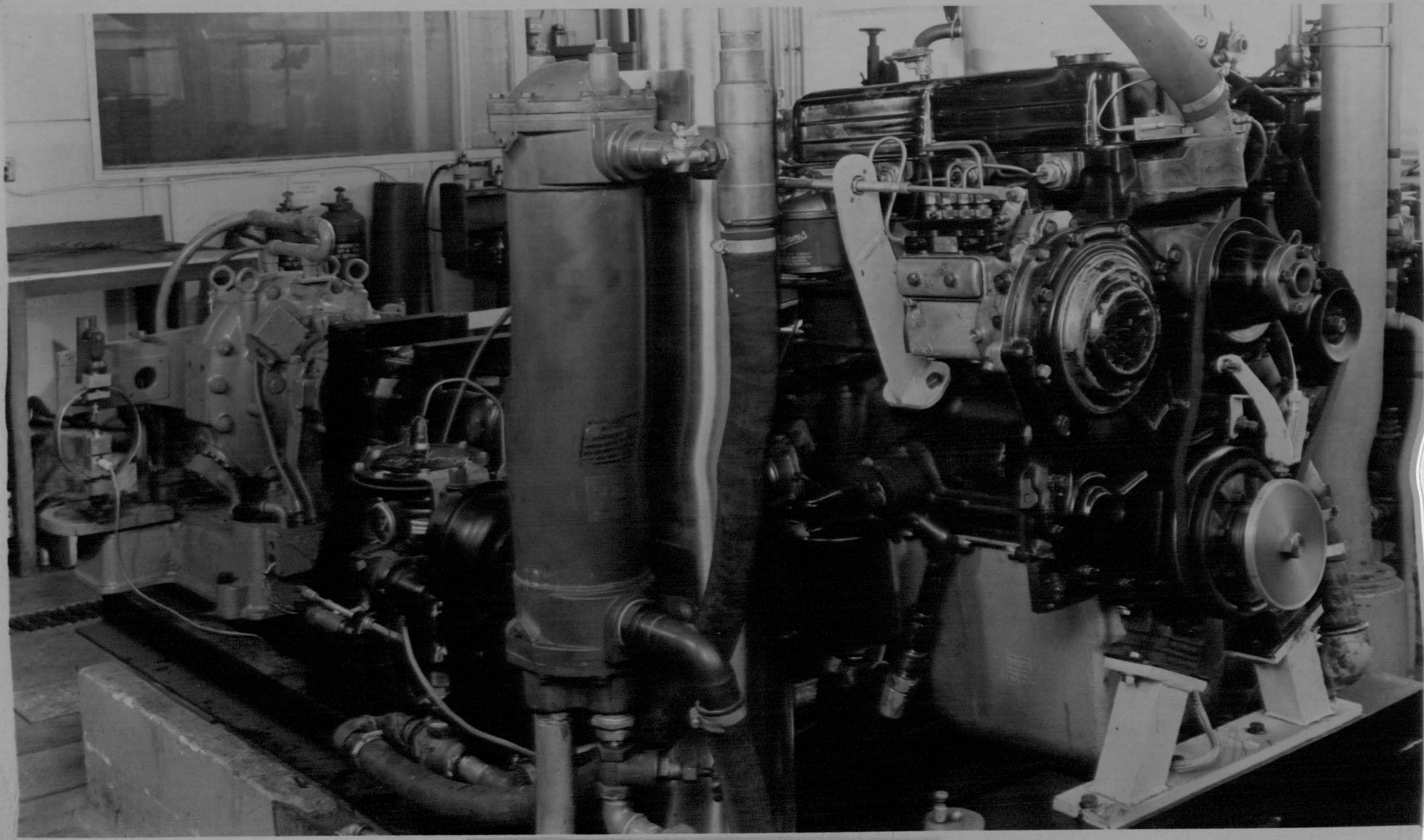


FIGURE 3.7 - DIESEL ENGINE TEST RIG

FIGURE 3.8 - PETROL ENGINE TEST RIG

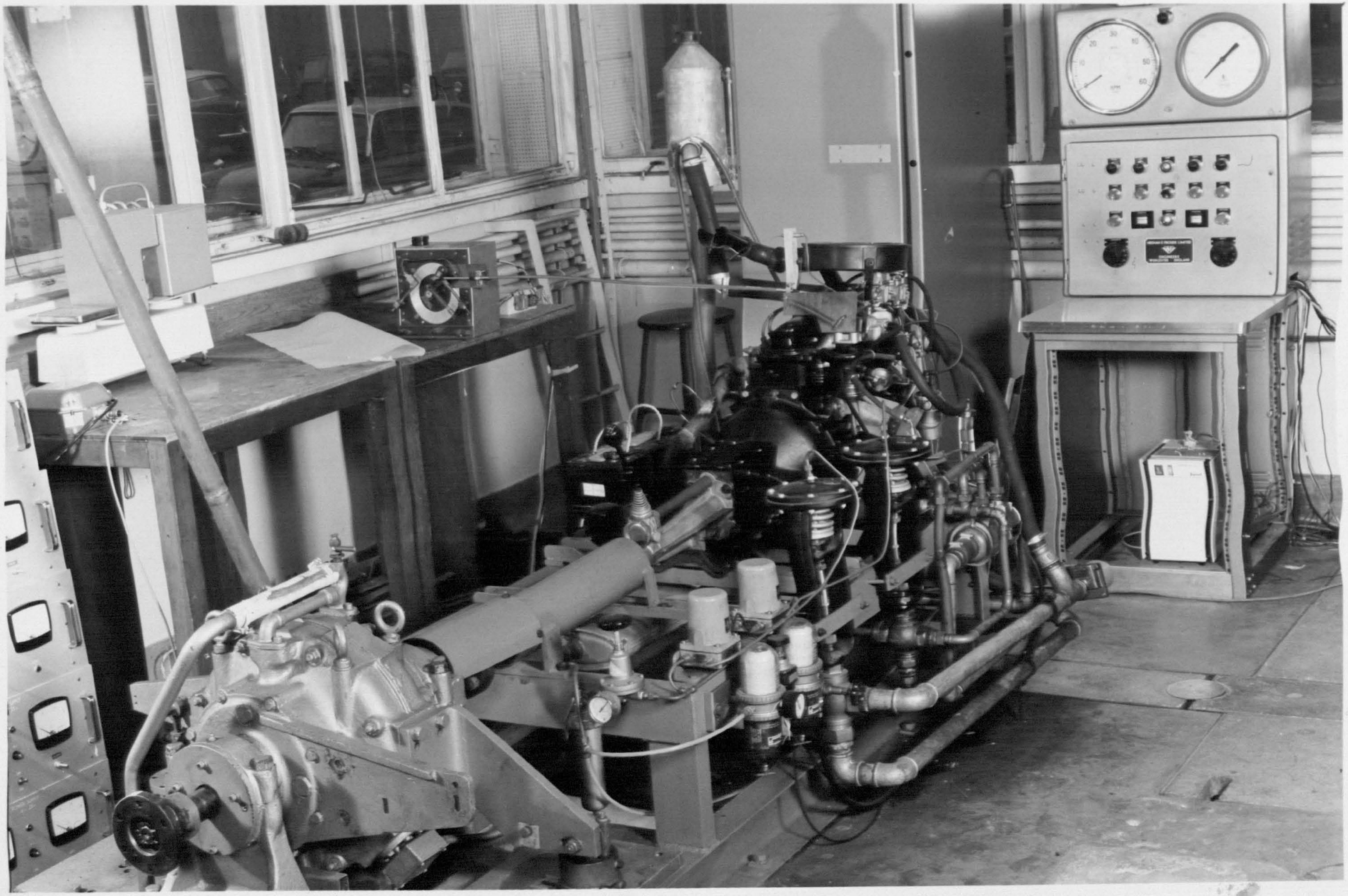
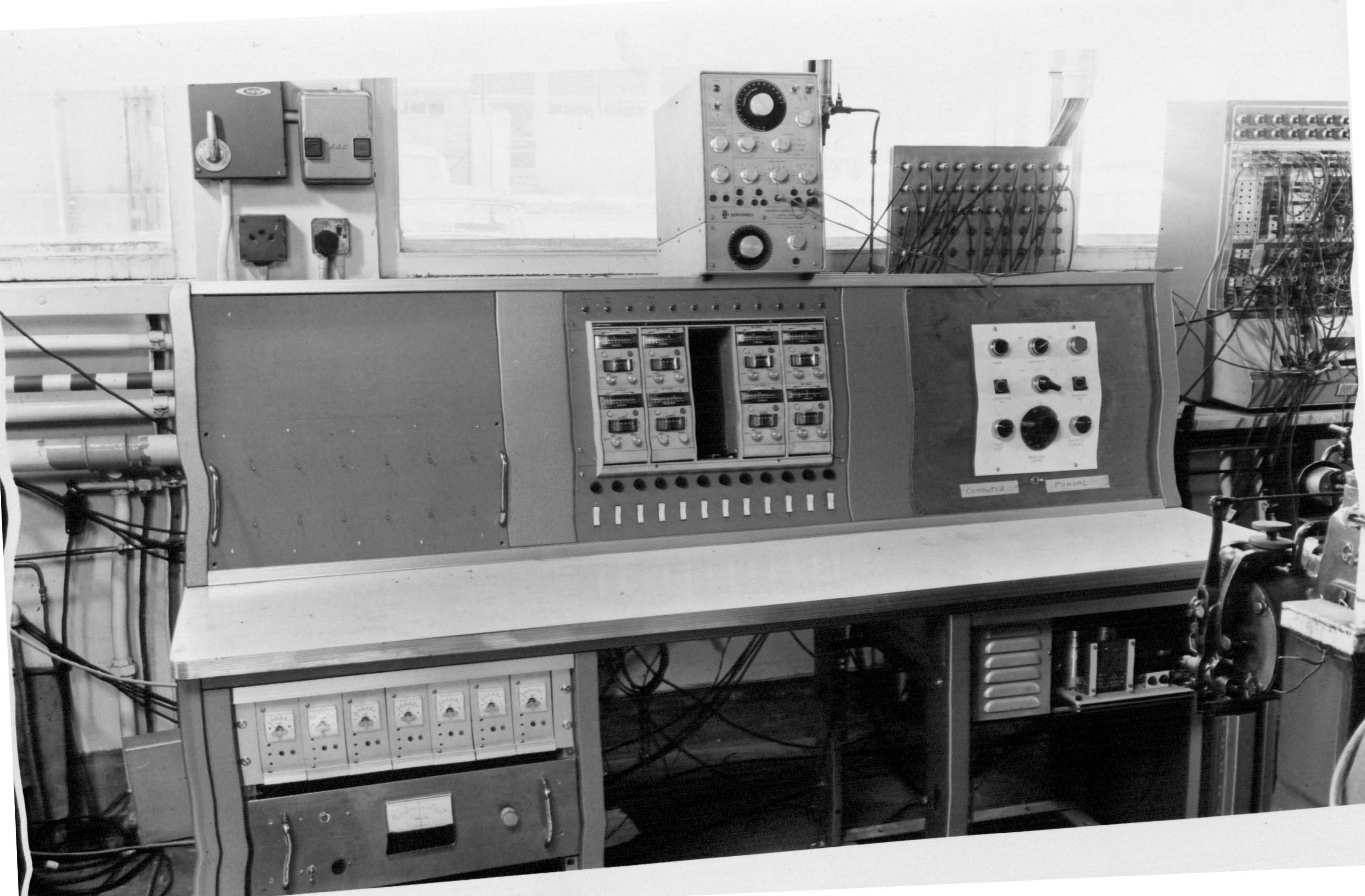


FIGURE 3.8 - PETROL ENGINE TEST RIG

FIGURE 3.9 - TEST RIG ELECTRONICS


COMPUTER
MANUAL

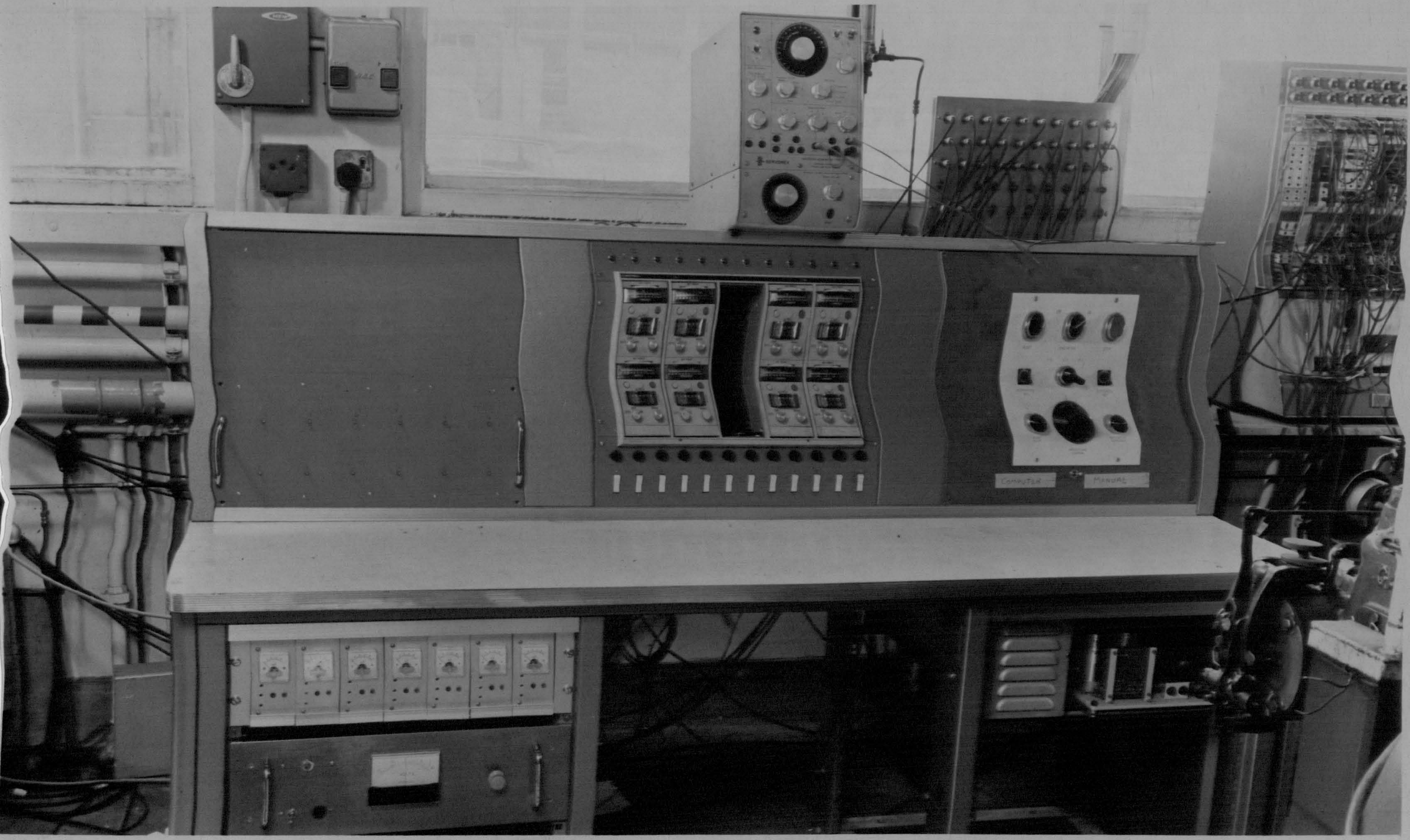


FIGURE 3.9 - TEST RIG ELECTRONICS

FIGURE 3.10 - DIESEL RIG ELECTRONICS


HONEYWELL
HOURS
SET POINT
PROCESS
OUTPUT
COMP
MANUAL
AUTO
POWER SUPPLY
6v 25A
ADJUST VOLTS
10A
OFF
ON
RESET
POWER SUPPLY
6v 25A
ADJUST VOLTS
10A
OFF
ON
GANG
POWER SUPPLY
6v 25A

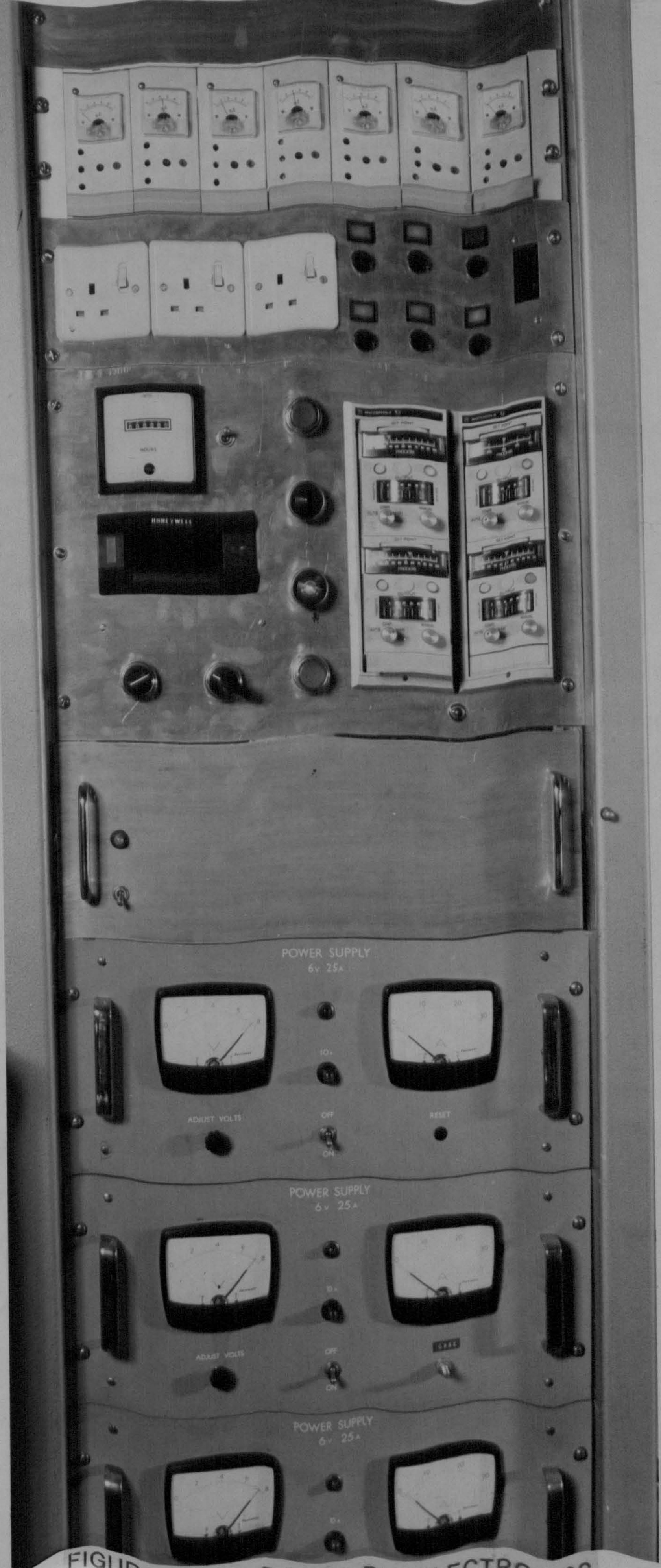


FIGURE 3.10 - DIESEL RIG ELECTRONICS

There were also a great many other secondary parameters that could be sensed by the computer system. A list of all the various types of data and the sensing method is given in Figure 3.6. The actual test rigs themselves together with some of their associated equipment are shown in Figures 3.7 to 3.10.

3.4 Dunton Test Beds

The test bed facility described in the previous section was used as the basis for a larger scheme built at the Ford Motor Company's Dunton Research and Engineering Centre. One of the test beds in this facility was in turn used for some of the experimental work included in this thesis. As a result a description of the facility is included here rather than in the previous chapter. Furthermore as a result of similarity between the two facilities, this section will concentrate on the difference between the systems. A fuller description of the Dunton facility may be obtained from the relevant references (11, 61, 62).

The overall structure of the facility is shown in Figure 3.11. The system consists of six test beds in individual cells of identical design. A further four cells with only minor differences in instrumentation also exist. All the test cells are based around local analog control systems and eddy-current dynamometers of the same type as those used at the college. The torque measuring system for all the cells uses a load cell connected to the dynamometer carcase. The electrical signal is digitalised and displayed by local instrumentation. The original six identical test cells are connected by a

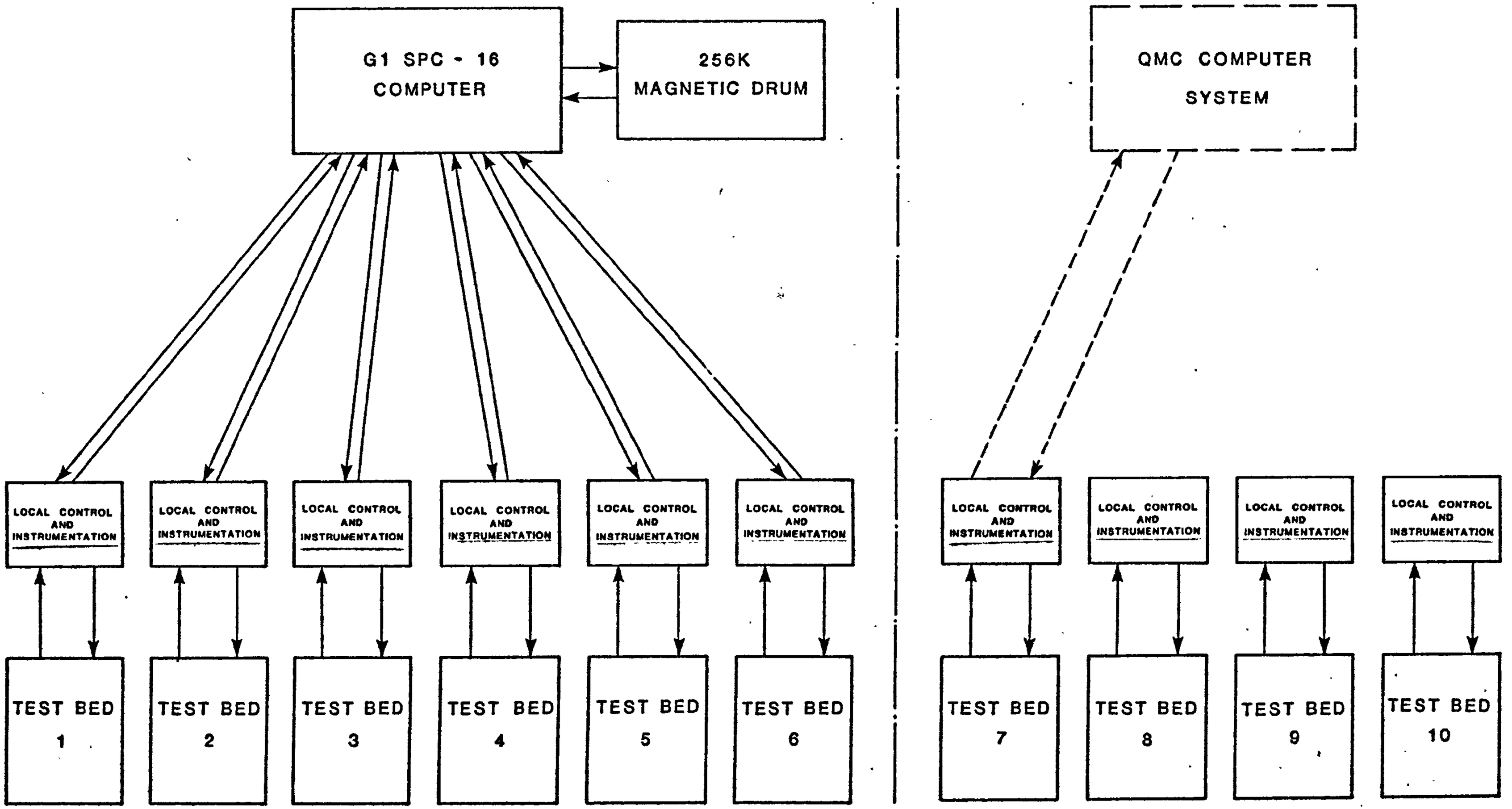


FIGURE 3.11 - DUNTON TEST FACILITY

General Automation SPC-16 computer which performs real-time multiple control of all the test cells, running a variety of standard steady-state type tests such as full load power curve and durability routines under the supervision of General Automation standard real-time operating system RTOS-16. Although the college system was able to perform multiple testing on its two test rigs simultaneously using only core memory, the increased requirement for six operational test beds has led to a magnetic drum type memory storage unit being added.

Although the additional four test cells are not connected to the computer, they have all the necessary circuitry to do so installed. Thus for the purposes of our testing it was only necessary to connect the test cell temporarily to another similar computer system.

Of course it was not possible simply to use exactly the same programming and hardware as was used at the College. The control of the test cell is somewhat more complex with a total of 10 digital lines being controlled by the computer DDOs to operate it. The test cell also includes a more comprehensive hardware emergency shutdown system which automatically alters set-points to the local controllers as well as activating the hydraulic shutdown equipment.

All ten test cells are used solely for diesel engine work and are equipped with AVL System 700 Fuel Consumption Measuring Equipment (7). The six computer controlled cells all have AVL 412 Smoke Meters (8) and an additional one was subsequently installed in the new test cell used in connection with the author's work. The amount of display equipment to allow manual working of the test bed is considerably more than that used at the college (see Figure 3.12).

FIGURE 3·12 - DUNTON CONSOLE

The test cells themselves are individual rooms housing a single engine and dynamometer test rig (Figures 3.13 and 3.14). The operator consoles are on a common corridor with a separate computer room for the main computer system. The College computer however was situated in the corridor outside the test cell to which it was connected.

3.5 Summary

The chapter provides a description of the two test systems that formed the principal experimental facilities for the work described in the subsequent chapters.

The common form of automation and control philosophy used in both cases is outlined and its implementation using control loops based on local analog control units described. The remainder of the test cell hardware is detailed in such a way as to provide a background for the sections describing the actual work on dynamic testing.

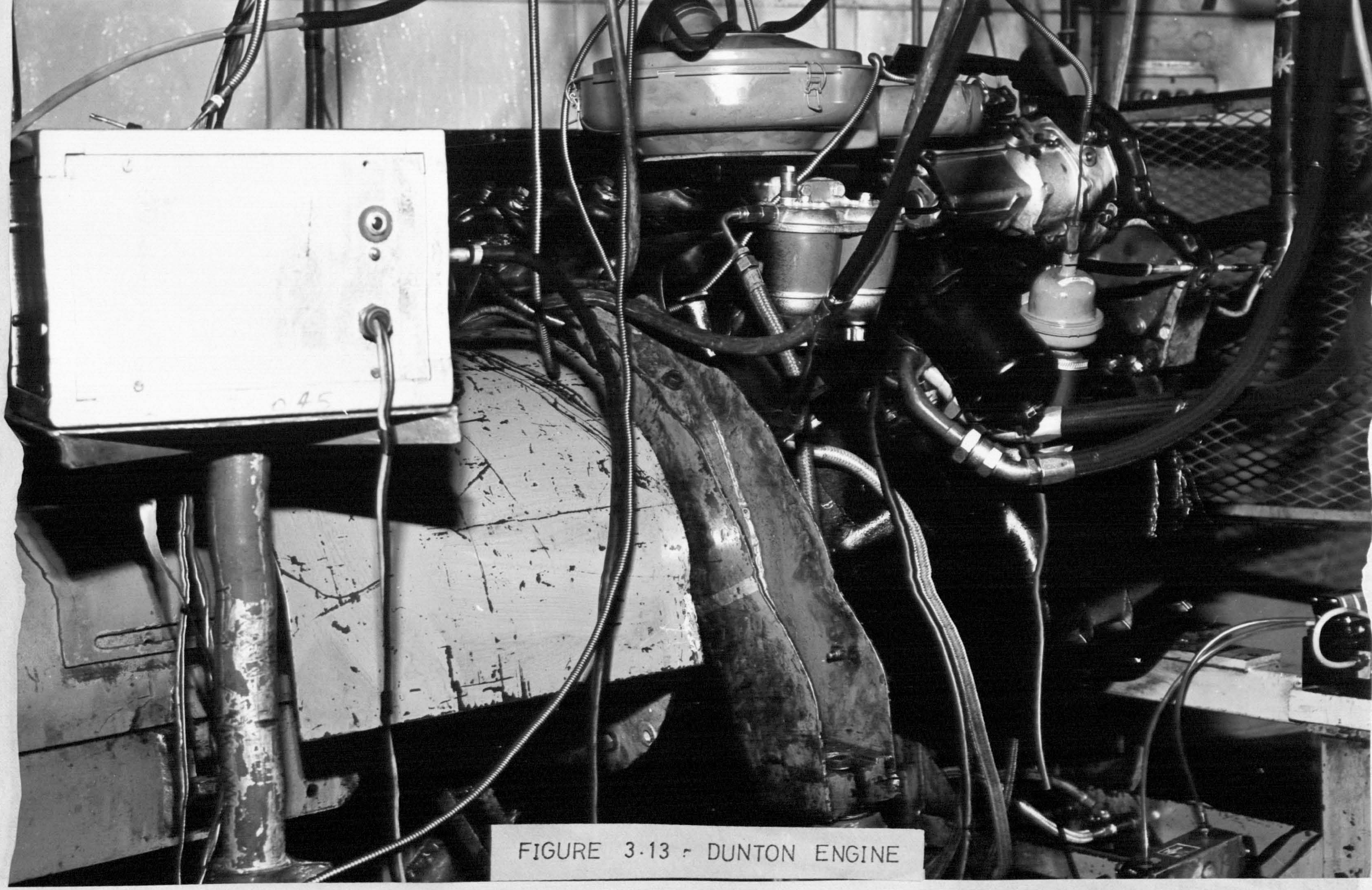


FIGURE 3·13 - DUNTON ENGINE

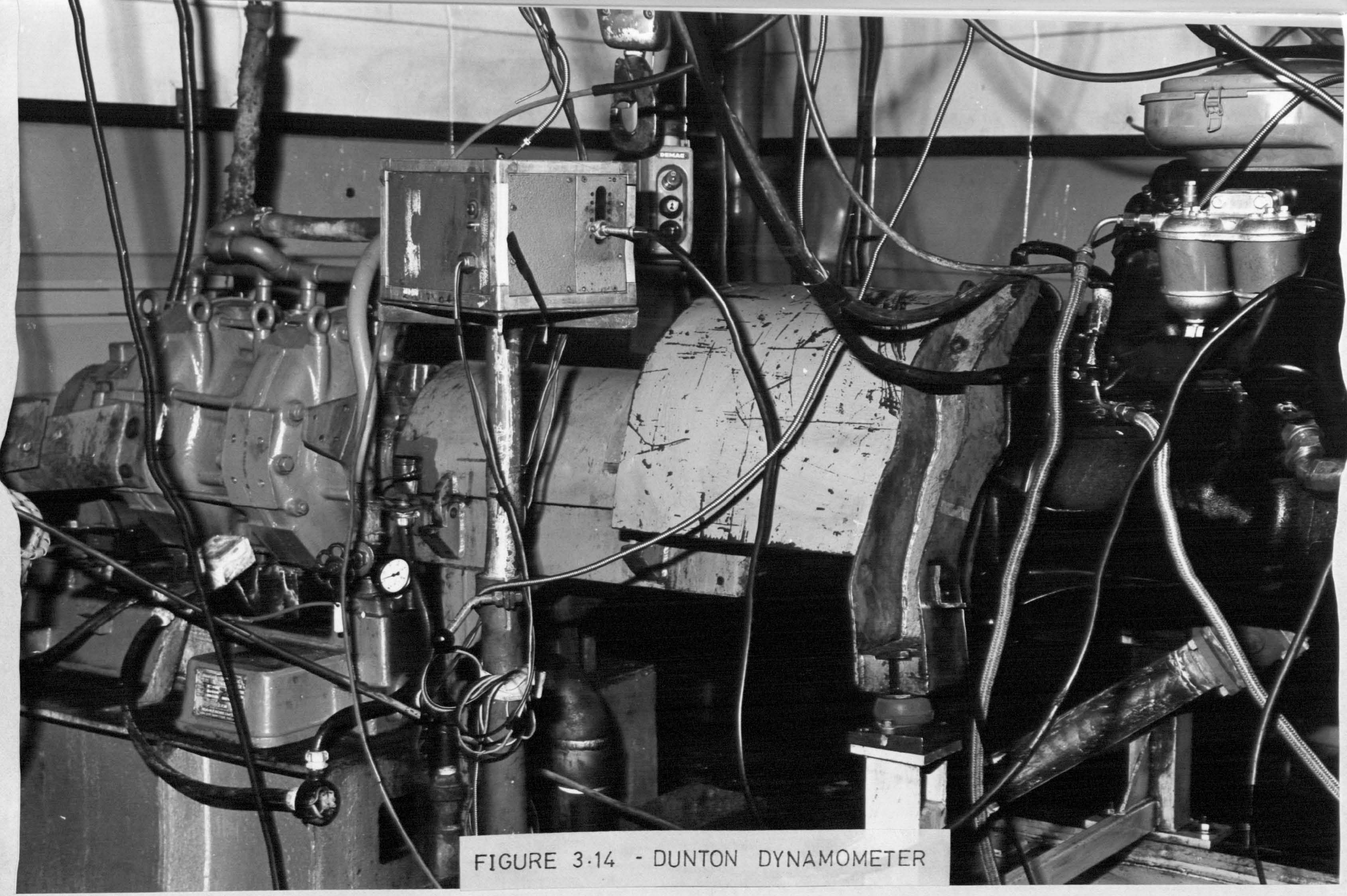


FIGURE 3.14 - DUNTON DYNAMOMETER

Chapter 4

PRODUCTION AND DEVELOPMENT TESTING METHODS

4.1 Introduction

Having set the scene by describing the hardware used in the test rigs, we can now turn to the various testing procedures which might be required either in a production or a development testing environment. Those described here are largely conventional in nature and have been in use in the automotive industry for many years. Some of the more modern and more specialised tests which are being introduced are dealt with in later chapters.

Perhaps the commonest test in use is the full load power curve, described below, which can be applied in both development and production testing fields. Later on in this chapter we will see how an automated version of this test can be implemented on the test rigs described in the previous chapter.

4.2 Test Procedures

The test procedures included here are intended to provide a sample of the type of work typically carried out in an engine test facility. The procedures are divided into two types, performance and diagnostic. Whilst a performance test can be used for fault testing, as a rule it will only give an overall indication of engine behaviour. A diagnostic test on the other hand is a special procedure which will not only show any abnormalities in engine performance but also provide some degree of indication as to the cause of the misbehaviour.

4.2.1 - Performance Testing

The most important division within this field to be considered is between hot testing and hot run procedures. In the case of hot run tests the engine runs under its own power but is not loaded in any way. The usefulness of such testing methods is limited as far as performance testing is concerned (although they play a major role in diagnostic testing) as the amount of data that can be obtained on the engines behaviour is small. However the hot run test technique of running the engine at a variety of speeds at no-load provides a method for production check-out and also a way of performing some limited amount of quality audit testing.

The greater part of performance testing involves loading the engine, usually by the use of an engine dynamometer,although some tests are performed using a chassis dynamometer. The tests normally consist of a series of steps. At each step a particular combination of speed, load or throttle position requirements are set up and the engine allowed to stabilise. The data parameters that are required to establish the engine performance are then logged and the test proceeds to its next step with new set points being used. This form of test is used in a wide range of circumstances and there are many special variants where other factors such as fuel mixture strength, or ignition angle, are also varied to provide additional performance data. Rather than attempt to cover all the different tests, a small selection of tests that play an important part in the work connected with this thesis are described below: -

1) Full-load Power Curve - This test is performed on both diesel and petrol engines to establish their behavioural characteristics over their normal operating speed range whilst developing full load at all times. The normal procedure is to bring the engine to its minimum operating speed with fully open throttle by applying dynamometer load. When all the parameters to be measured have stabilised, the data can be recorded. This procedure is repeated for a series of steps in engine speed until the maximum rated speed of the engine is reached. The size of the speed increment per step, varies according to the amount of data required. In a typical development test environment this step would be 100 to 200 rev/min.

2) Part-load Power Curves - In the case of this test, the torque should be controlled to some constant value during the entire duration of the test. A series of speed settings similar to those for the full-load curve are used with data parameters being logged at each point. The test may often be repeated for several different values of torque interspersed between no-load and full-load.

3) Governor Curves - These apply to diesel engines only and are used to map the behaviour of the governor in the engine fuel pump. The engine throttle is opened fully and load applied via the dynamometer until the engine speed is slightly below the nominal cut-in speed of the governor. The load is then reduced in small steps to allow the speed to increase. The setting of the dynamometer load is done directly by the use of the Propeller Law Mode.rather than via any

control loop, so that change in load and speed is always uni-directional rather than allowing overshoot and correction to occur. After each step, the torque and speed are logged and this continues until the residual or no-load condition is reached (see Figure 4.1). The reverse curve is then performed in the same manner to allow the hysteresis effect to become apparent. The results will also show the actual cut-in speed and the run-out percentage which is the ratio of cut-in speed to the no-load speed.

4.2.2 - Diagnostic Testing

The majority of diagnostic tests are basically of the hot-run type where the engine is run without any form of load. In addition the actual control of the engine speed is not usually critical. The most complex area of the test is often the special instrumentation and analysis equipment which needs to be used.

1) High friction and low compression test - This is a 'cold' test where the engine is not run under its own power. An external source is used to turn the engine over at a slow speed (sometimes the engine's own starter motor can be used) and the torque exerted by the driving motor is monitored to provide an indication of the engine torque input to crankshaft angular position characteristic. A high level of average torque is usually indicative of excessive bearing friction. The shape of the torque characteristic will show low compression on any given cylinder.

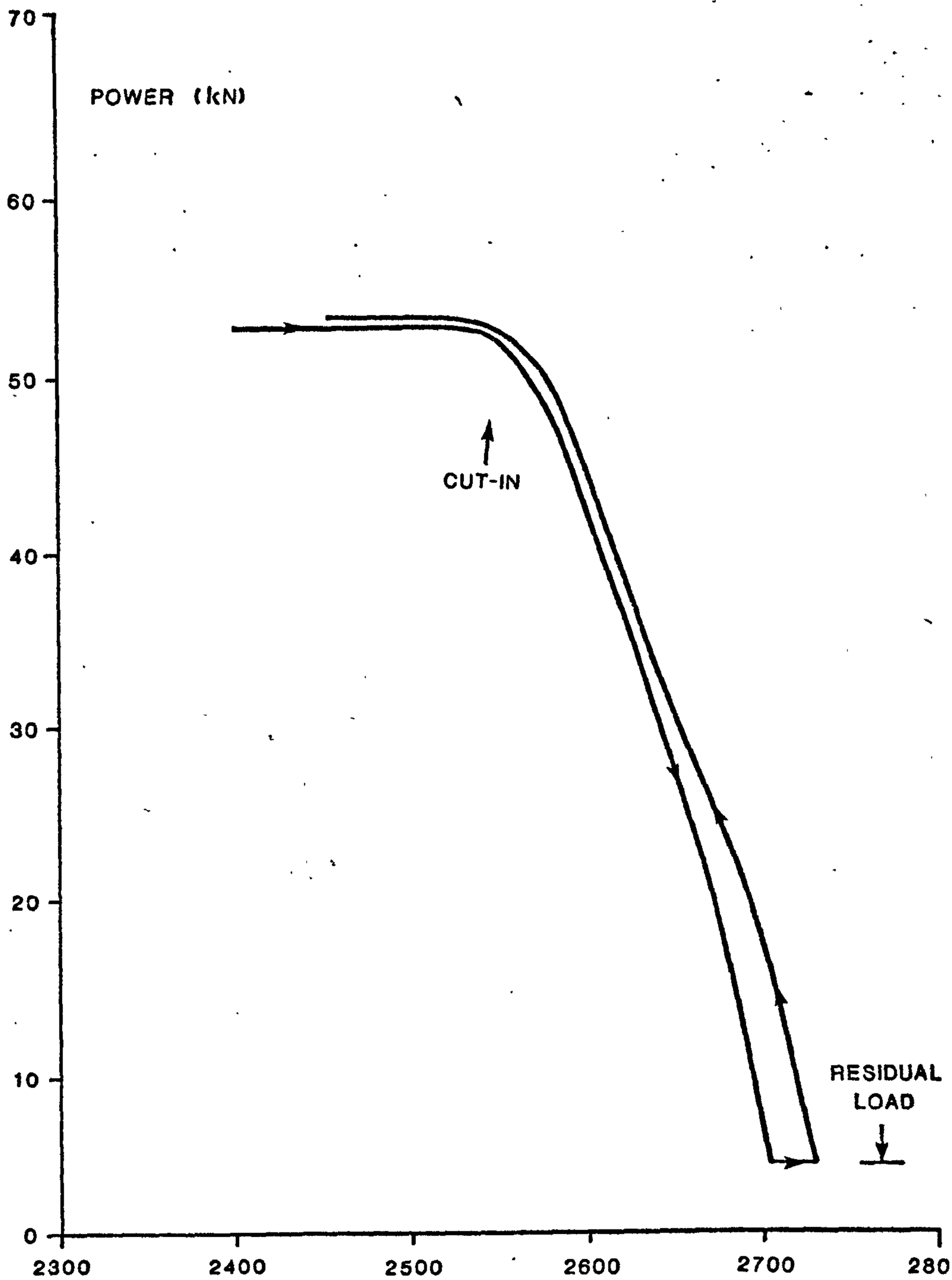


FIGURE 4.1 - GOVERNOR CURVE EXAMPLE

2) Power contribution test - Sometimes called the morse test. The contribution made to the total power by each cylinder is evaluated by disabling one cylinder at a time. If the engine is run at a constant throttle position with no load, then each cylinder disablement should result in an equal drop in engine speed. A lower than normal drop indicates malfunction in that cylinder. Alternatively the engine can be run in Mode III control, and the torque developed with each cylinder disabled measured.

3) Delta speed test - The deviation from the mean speed is measured for one cycle of the engine. The delta speed/crankshaft position shape can provide similar information on each cylinder to the power contribution test above. Whilst requiring fast data acquisition this technique has the advantage that it does not need any control or alterations to the engine.

4) Gas analysis testing - In addition to its use for measuring exhaust emissions for environment or legislative reasons (see Chapter 6), gas analysis can also provide information on engine performance and malfunctions (49). A particular use of this is related to the operation of the carburettor and ignition systems in petrol engines.

4.3 Conventional Power Curve Automation

The following section deals with the way in which a software package was written to provide the ability to run full-load power

curves using the computer to replace the manual actions of the human operator. This represents the basic initial approach to engine test automation with the use of the computer to set up controlled steady-state conditions, log the necessary data and perform various calculations before outputting the required test results.

The software package was written in Fortran IV using the General Automation SPC-16 for use on the test rigs at the College described in Chapter 2. Later on some parts of the package were modified to enable its use in connection with the dynamic power curve package at Ford's Dunton Research and Engineering Centre. The primary use of the package was to provide a yardstick against which the dynamic tests could be compared.

The algorithm of the package master program is shown in Figure 4.2. The first stage of the test is an operator/machine dialogue to establish the various parameters necessary for the test. These include the start and finish speeds of the test and all the intermediary logging points and the barometric pressure (used to calculate corrected torque and power). The speed for each logging point is stored in an array called SPEED and the total number of steps as a variable called LAST.

Before commencing the test the computer must take over control of the various test rig set points from their local settings by sending the computer control mode signal via its digital outputs. Before doing this it checks that the engine is running (speed greater than 500 rev/min) but that it is not running too fast (speed less than 1100 rev/min). The subroutine START (Figure 4.3) checks both

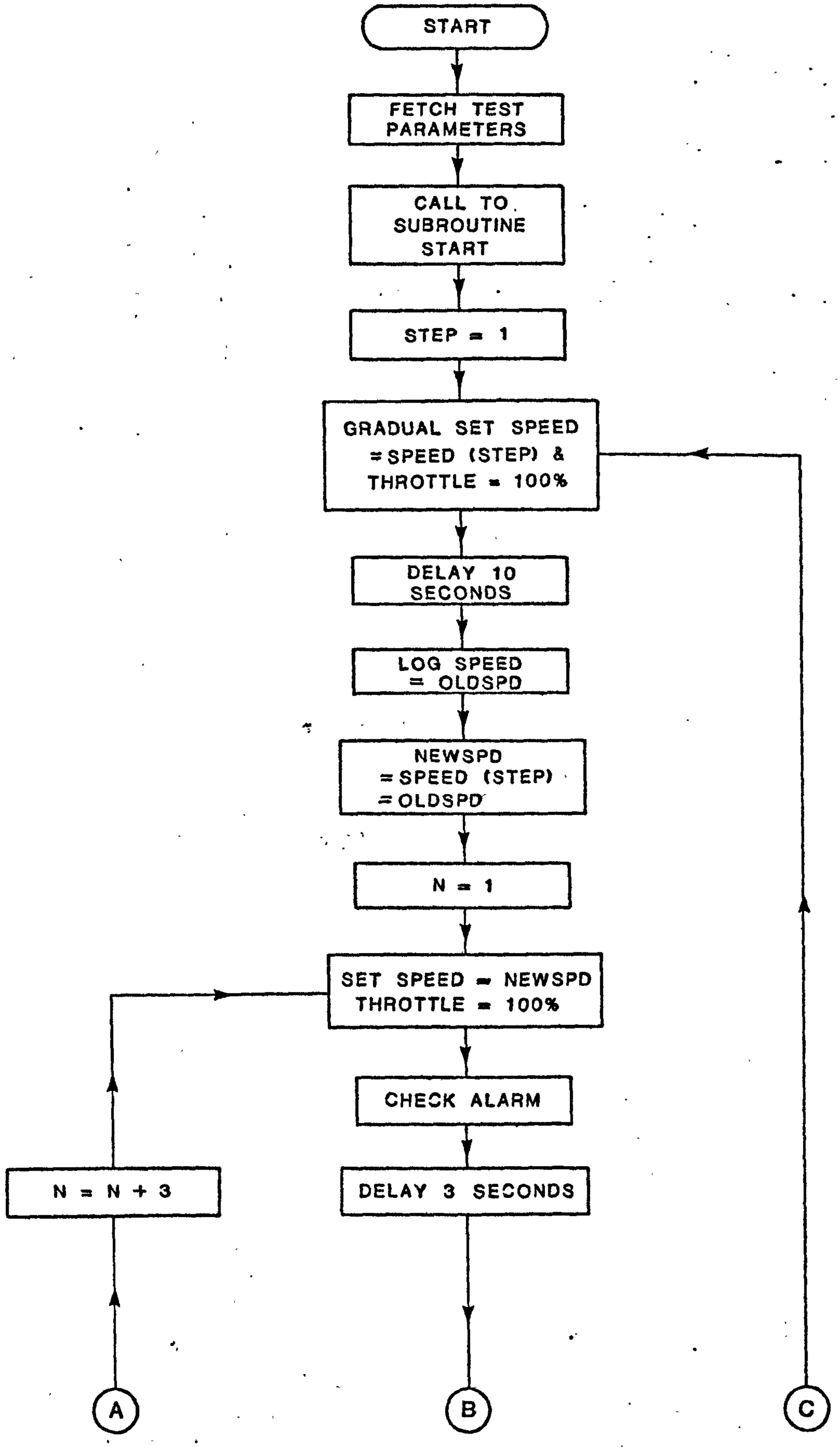


FIGURE 4.2 - POWER CURVE ALGORITHM

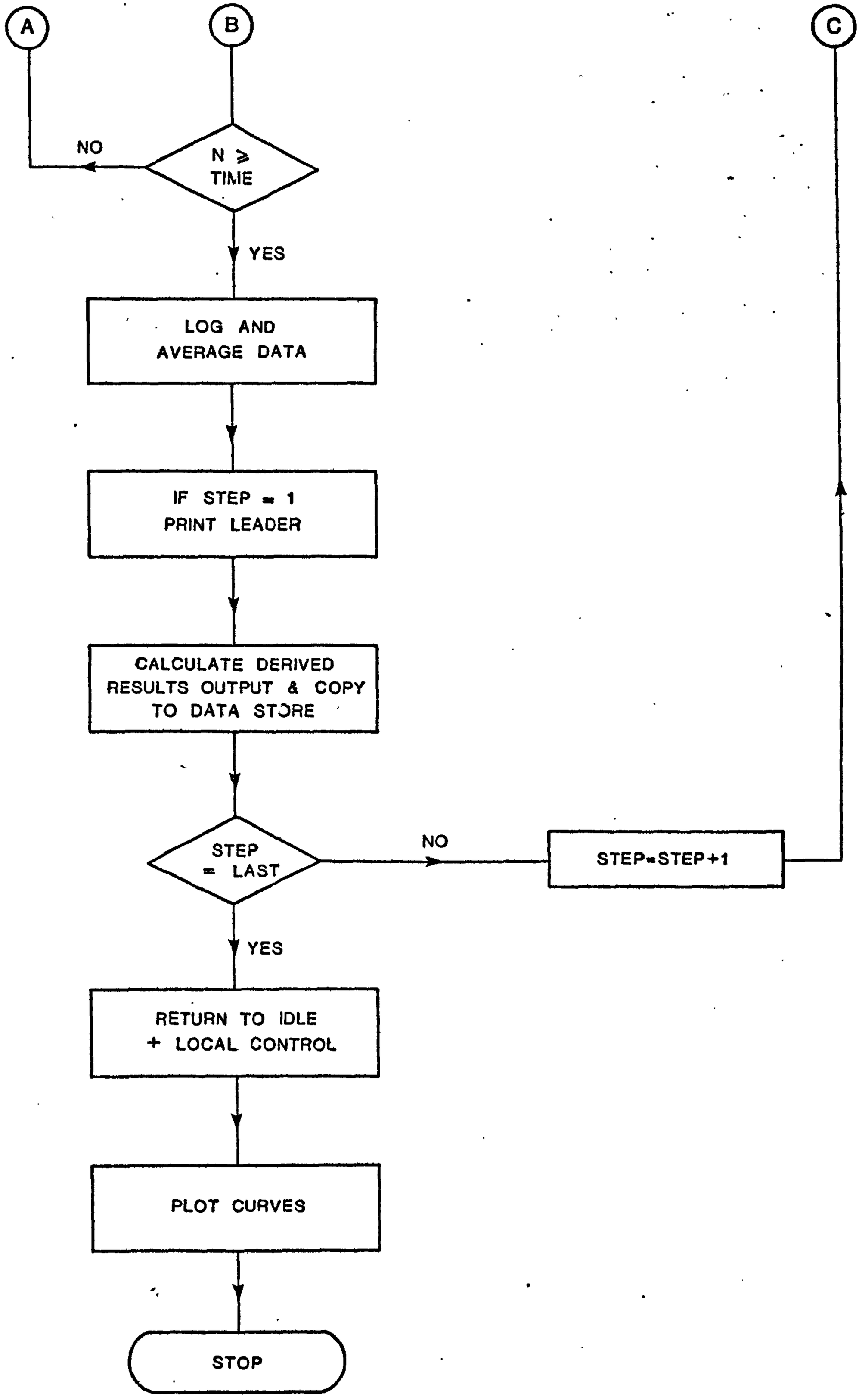


FIGURE 4.2 - (CONT.) POWER CURVE ALGORITHM

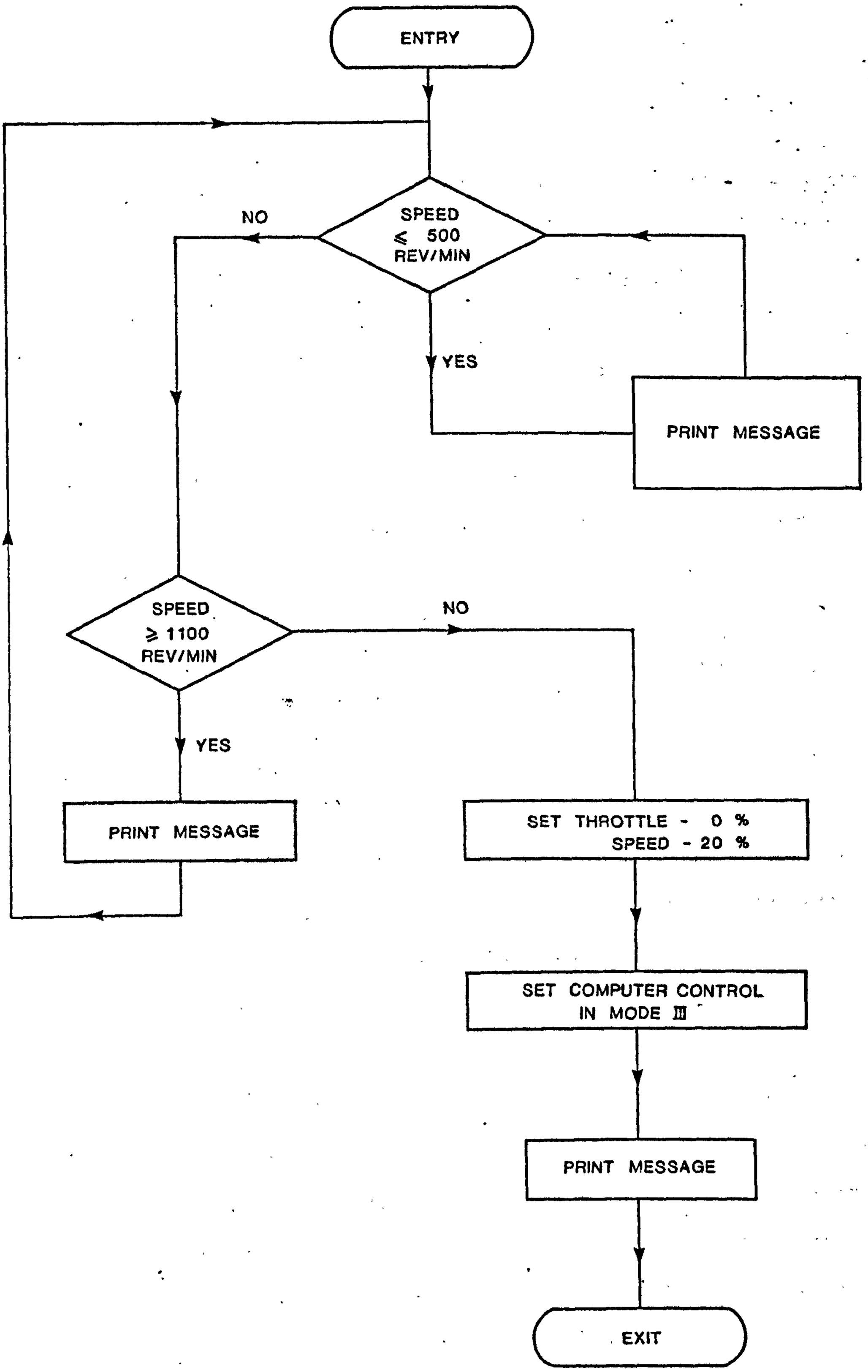


FIGURE 4.3 - START ALGORITHM

these criteria and, if they are not met, prints the appropriate message on the operator teletype (i.e. SPEED IS HIGH or SPEED IS LOW). The test procedure will not continue until the engine speed is within the required bracket.

When the computer decides that the engine is ready it will set its own analog output set point lines to request fully closed throttle and a speed demand of 20% of full scale speed. This will ensure that the engine will come to idle conditions without any load being applied when the computer outputs the command to take the control of the test rig via its digital outputs. When computer control is achieved the message COMPUTER CONTROL O.K. is output to the teletype.

There are in fact two different ways in which the programmer can alter test bed set points within this particular package. The first is by a call to a subroutine SET with arguments representing the speed in rev/min and the throttle opening expressed as a fraction such that unity represents a fully open throttle. This call will result in the set point outputs being immediately stepped to their new values. Although this technique works well with small changes in demand its usage is not to be recommended where the alterations are large. Sudden alteration from say fully open to fully closed throttle combined with some large alteration in speed set point can have a disasterous effect on the test rig with problems such as propellor shaft overload and failure. As a result of this there is an alternative set point subroutine called GSET . This routine has the same arguments associated with it as SET but will automatically break any large alterations in a set point down into a series of smaller steps

spaced out over a short period of time. Initially the step limit was 500 rev/min for speed and 20% for throttle position, with an interval between steps of 3 seconds. Thus if the engine was for example running at 25% open throttle and 1700 rev/min, and a call was then made to GSET requesting full open throttle and 3000 rev/min speed, the following alterations to the set points would be made by GSET before return from the subroutine call:-

Time (seconds)	Throttle Set Point (%)	Speed Set Point (rev/min)
prior to call	25	1700
to (call made)	55	2200
to + 3	75	2700
to + 6	95	3000
to + 9	100	3000
Returns to calling program		

This technique worked well on the test rigs at Queen Mary College but later on when working at Ford Dunton Research and Engineering Centre, was found to be still too severe on the system when using large, powerful diesel engines. As a result the subroutine was modified so that instead of using a few medium sized steps it would generate many small steps in rapid succession. The values used were maximum steps of 50 rev/min in speed, 2% in throttle opening and an interval between steps of 200 milliseconds. This resulted in the engine behaving very much as if it was responding to smooth demand ramps and there were no problems with dangerous sudden transients.

The way in which the actual test is run on the test rig using a loop procedure, can clearly be seen from Figure 4.2. The initial call to change the set-points, uses GSET to allow for large steps and to open the throttle at the start of the test. After allowing the engine a short period to settle, the actual engine speed is logged. The error between actual and desired speed is used to calculate a new set point. This outer digital speed control loop removes any off-set errors that may occur. When this has been done the program enters a loop where it refreshes the set points (to avoid analog out-put drift), checks for overspeed alarm condition, and delays for 3 sec-onds before repeating. Unless an alarm situation occurs, in which case the test is aborted and the computer enters an emergency shut-down routine, it will continue executing the loop until a pre-defined time has elapsed. In the earlier versions of the package this time was set to 3 minutes and could only be changed on program modification basis. Later however this was changed to allow the operator to de-fine the duration of this settling period during test parameter fetch dialogue at the beginning of the test.

When the required settling time has elapsed, the computer will perform a logging operation. Because of the steady-state nature of the test it is easy to use digital filtering to iron out any errors due to minor perturbations in the data. The routine AVLOG which is called at this stage cumulatively logs speed and torque signals so that a total of five readings are taken at one second intervals. These are then averaged to provide the final results. The rest of the variables such as temperature and pressures are reasonably stable

and hence are logged by a single direct reading of the relevant analog input channel. The program was later complicated somewhat by the additional requirement to log data from an AVL Smoke Meter. This device required an initiation signal from the computer's digital outputs, followed by an interval of about 15 seconds, during which the actual exhaust sampling took place, until the smoke reading became available as an analog voltage. To operate the smoke reading system the AVLOG program was modified so that its first action was to send the initialisation signal, followed by the speed and torque logging and averaging operation that took about 5 seconds to perform including the single loggings of the other data parameters. A delay statement then held up execution of the program for 12 seconds after which the smoke reading would be logged.

For the first speed in the test procedure the computer would print out the header for the test data on the teletype. Following this (or immediately after the logging in the case of subsequent loops), the computer entered the data output phase. This consisted of calculating the BMEP and power output and performing correction of results for atmospheric temperature and pressure variations. The results would then be printed out onto the teletype and also those variables which would be required for graphical plotting copied to a data storage area in memory.

With this the step would be completed and the computer could return to the start of the loop to move on to the next logging point. When all the logging points were dealt with the test was finished. The engine was brought back to idle by a call to GSET to close the

throttle fully. When this was complete the test rig could be released from computer control. The final part of the test was to plot the curves of the corrected values of torque, BMEP and power against speed. The subroutine which performed this operation automatically calculated a series of interpolation values to plot straight lines between successive logged points. The values calculated were outputted to an X-Y plotter via two of the computer analog outputs and one of the digital outputs was used to control the raising and lowering of the plotter pen.

4.4 Other Steady State Test Automation

Although the package above is specifically intended for full-load power curve tests, it also provides basic facilities to allow anyone capable of writing simple Fortran programs to generate other steady state test procedures. All the difficult work of interfacing with the test rig is done for him by one of the various special library routines. To see what this means consider that the would-be programmer has access to the following routines:-

1) AVLOG - Performs an average log of speed and torque and a single scan of other data inputs from the test bed.

2) LOG - Results in a 'single-shot' logging of speed and torque.

3) SET - Set the speed demand and throttle position to the specified values directly.

4) GSET - Gradually ramp the speed demand and throttle position to the specified values.

5) CHECK - Check engine for overspeed and if true, shutdown rig, print alarm message and abort test.

6) DELAY - Hold execution for the specified time duration.

7) MODE - Take over or release computer control of test rig depending on argument.

8) CORR - Calculates corrections for variation in atmospheric pressure and temperature.

9) PLOT - General purpose graph plotting routine

The actual operation of the routines will depend on the arguments transferred during any subroutine call and on variables in common storage (such as the maximum permitted engine speed). Now the only demands on the programmer are only to be able to use various standard FORTRAN statements. This means that the programmer need only have a knowledge of scientific or calculative type programming, and will be able to generate steady-state packages without knowing or understanding the details of control or systems programming.

4.5 Summary

In this chapter a representative sample of the sort of procedures that are generally in use in engine testing has been presented. This is intended to show that the great majority require no more than steady-state control abilities. Continuing from this point, the way in which one of the most common of performance tests can be auto-

mated is shown. This has been developed into a concept that provides a basis for automating most conventional steady state tests. Although limited in its abilities from a dynamic point of view the resultant techniques provide a speedy way of performing tests coupled with the ability to perform the data processing immediately and give direct output of calculated results thus providing a powerful tool for the engine testing.

Chapter 5

DYNAMIC TESTING AS A REPLACEMENT FOR CONVENTIONAL OR STEADY STATE TESTING

5.1 Introduction

In the previous chapter we have seen how an automation strategy may be applied to improve many of the facets of our engine testing work. This, however, is only the first stage. Although our primary objective of replacing manual or semi-automatic operation by the use of a digital processing system, can in itself justify its own installation, we should not limit our thinking to this aim alone. The next stage leading on from our initial system designs, is to consider in what ways we can use our automated test systems to perform testing techniques that would not be at all practicable in a manually orientated environment, rather than simply automating our original manual operations.

Perhaps the most significant facet of most conventional testing procedures is their use of steady state conditions. This, as in the case of the conventional full-load power curve, involves allowing the engine to stabilize at a fixed speed, and after a delay taking a log of the various data parameters required before moving on to the next point. Although many arguments have been presented to justify this arrangement, such as the requirement to allow internal engine temperatures to stablize, it is felt that these are perhaps given unjustifiable importance and that a far more significant cause in the past was the actual technique by which data was logged. When each of the many data items has to be read in turn by the operator and then copied down by hand, it is obvious that the engine must be held at the particular state in question for a considerable period of time, whilst the readings are taken.

With the advent of automated methods of data logging, it becomes apparent that this constraint is no longer applicable. Here is an area in which we can use our automation techniques to improve on conventional methods, by performing our tests dynamically rather than by steady state methods. Figure 5.1 illustrates this. In, for example, a situation where we wish to take a series of data readings at 200 rev/min intervals between speeds of 1000 and 2000 rev/min, instead of setting the engine speed to each value in turn and then allowing the engine to settle for a period before logging the data, we could allow the speed to follow a ramp between the first and last speed points. If our data logging ability is practically instantaneous, then the data at each of the required points can be logged as the engine passes through that speed.

The primary advantage in using the dynamic testing method lies in the fact that it can shorten the time taken to perform a test when compared to conventional steady state testing. The actual acceleration of the engine, the ramp rate, will determine the duration of the dynamic test and more is said about this later. However when we consider that a settling time between stages of a steady-state test is often in the region of 5 to 15 minutes, even comparitively slow ramp rates can result in significant time savings. In this way, throughput per test stand is increased making our testing procedures more economical.

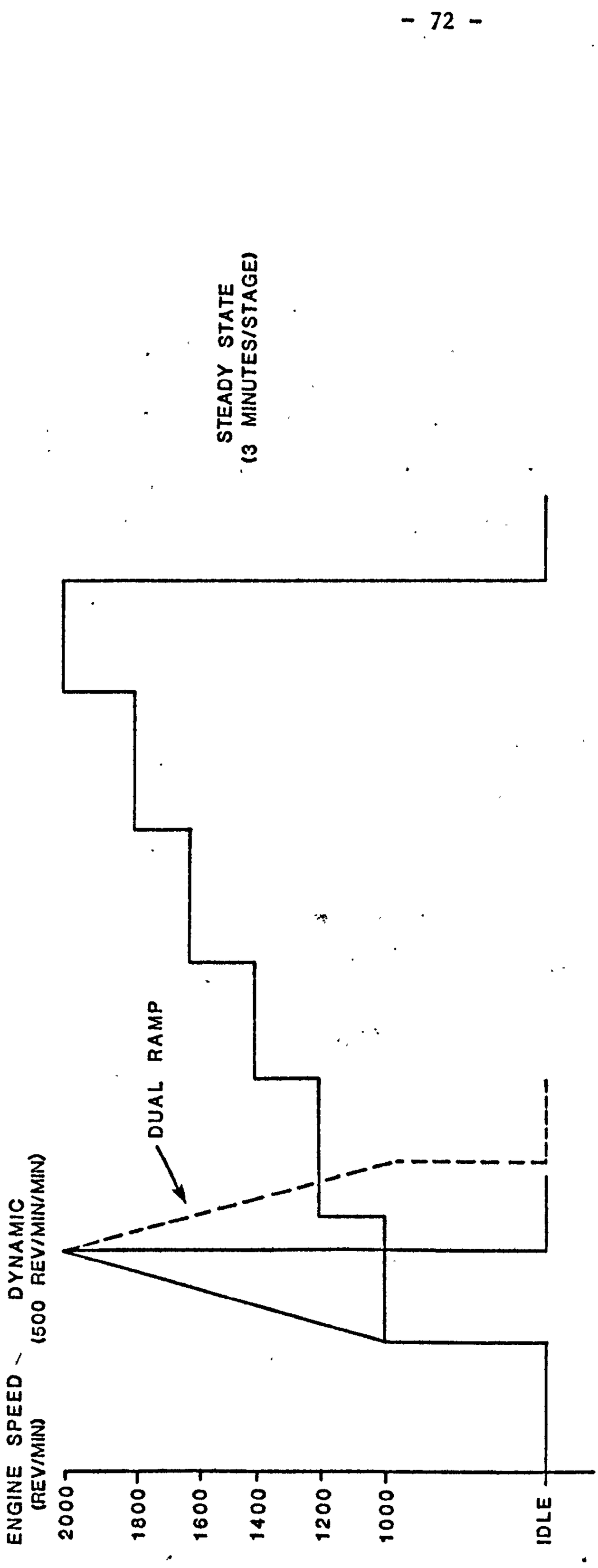


FIGURE 5.1 - COMPARISON OF DYNAMIC AND STEADY STATE DURATIONS

5.2 Theoretical Considerations

Before proceeding to the actual dynamic power curve, testing itself, there are some theoretical considerations that should be taken into account. In this section we must look at the way in which we will expect our dynamic results, to relate to comparable results obtained by steady state methods. Our prime concern is the relationship between the measured torque and the actual torque being developed by the engine. The commonest form of torque measuring instrumentation is the load cell measurement mounted on the dynamometer carcase (see Figure 5.2). In this case the torque developed is given by the following equation:-

$$Q_D = kX_{DC} + C\frac{dX_{DC}}{dt} + I_{DC}\frac{d^2X_{DC}}{dt^2} \qquad (5.1)$$

(Ignoring the torque absorbed by frictional effects)

where Q_D = Dynamometer torque

X_{DC} = Angular deflection of the carcase

k = Spring Constant

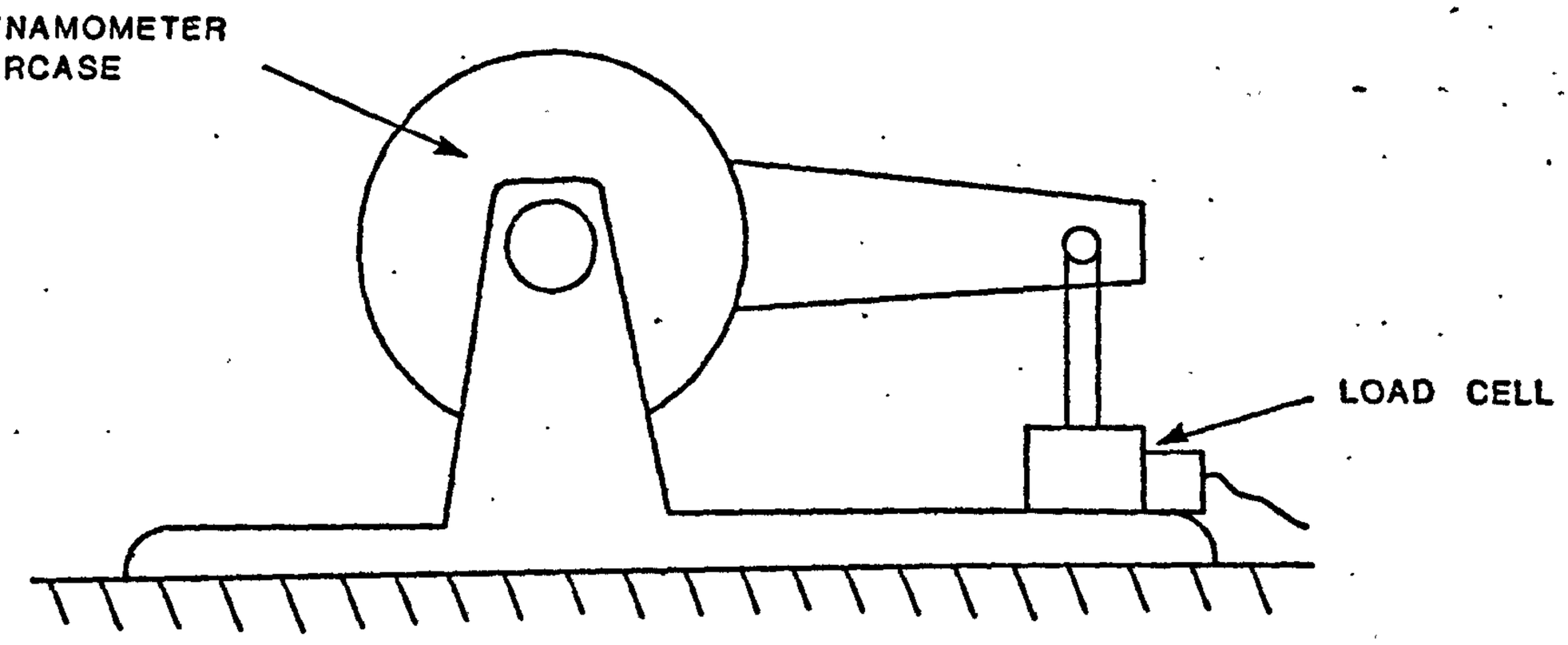
I_{DC} = Inertia of the Dynamometer Carcase

C = Damping Constant

On the other hand the torque measured by the load is given by:-

$$Q_M = k\,X_{DC} \qquad (5.2)$$

Where Q_M = Measured Torque



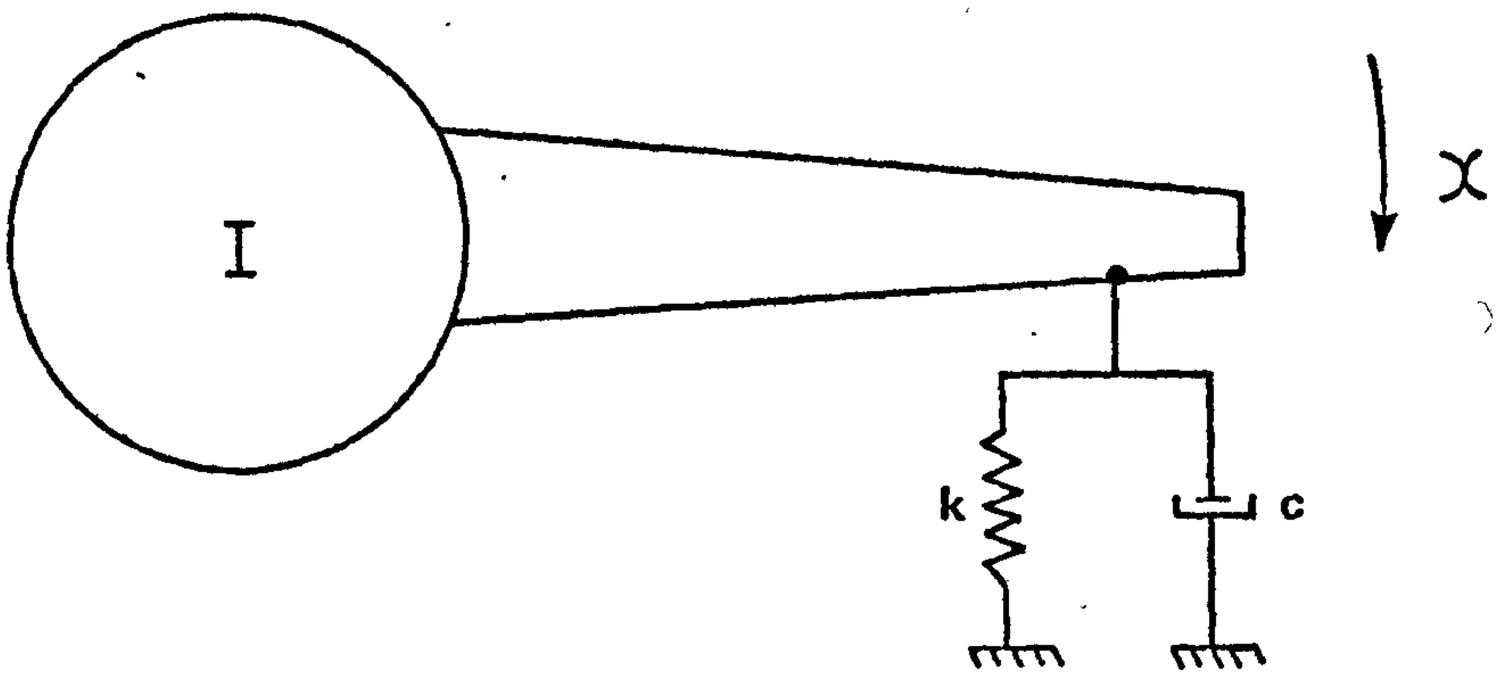


FIGURE 5.2 - INDIRECT TORQUE MEASUREMENT

Now if we consider the steady state case, the dynamometer carcase is at rest, thus:

$$\frac{dX_{DC}}{dt} = \frac{d^2X_{DC}}{dt^2} = 0 \qquad (5.3)$$

Hence from the above three equations we can say that, for steady state conditions:-

$$Q_{M_{ss}} = Q_{D_{ss}} \qquad (5.4)$$

If we turn our attention now to the engine and dynamometer dynamics, the overall rotational behaviour of the system can be described by:

$$Q_E - Q_D = I_{ED} \frac{d^2\theta_{ED}}{dt^2} \qquad (5.5)$$

where Q_E = Torque developed by the engine

Q_D = Torque developed by the dynamometer

I_{ED} = Moment of inertia of the total rotating system (dynamometer rotor, engine rotational parts and propellor shaft)

θ_{ED} = Angular position of the system

It should be noted that the above equation makes the assumption

that the rotational system can be considered as a single rigid unit, i.e. that the twist angle of the propellor shaft can be regarded as constant.

The second order derivative of θ_{ED} with respect to time is the angular acceleration of the system and in the steady state case this is equal to zero, therefore:

$$Q_{E_{ss}} = Q_{D_{ss}} \tag{5.6}$$

and from equation 5.4 we can obtain our validification of the normal steady state torque measuring system, i.e. that:

$$Q_{E_{ss}} = Q_{m_{ss}} \tag{5.7}$$

On the other hand in the dynamic case our various accelerations are no longer zero quantities:

$$\frac{dX_{DC}}{dt} \neq 0$$

$$\frac{d^2X_{DC}}{dt^2} \neq 0 \tag{5.8}$$

and $$\frac{d^2\theta_{ED}}{dt^2} \neq 0$$

and so for dynamic testing our equivalent equation for 5.7 becomes

$$Q_E = Q_M + C \frac{dX_{DC}}{dt} + I_{DC} \frac{d^2X_{DC}}{dt^2} + I_{ED} \frac{d^2\theta_{ED}}{dt^2} \qquad (5.9)$$

The result of this is that we can no longer assume that the torque measured by the load cell is the same as that being developed by the engine, and so we must consider more refined techniques to obtain an accurate figure for the torque.

There are three alternative approaches that could be used for obtaining a valid torque measurement. One of these was rejected outright as unfeasible in practice. Both of the remaining two were investigated as part of the development of the dynamic power curve.

5.2.1 - Constant Offset Method

This technique is at first sight perhaps the most logical approach. It can be shown that for a given test using a constant acceleration ramp, equation 5.9 can be simplified to:-

$$Q_E = Q_M + K \qquad (5.10)$$

where K is a constant dependent upon the acceleration of the system. This means that if we can evaluate K by experimental or theoretical methods, we can easily perform a correction to obtain the actual developed torque from the measured torque.

In fact this approach was not adopted as a feasible possibility for two basic reasons. Firstly, although our basic theory allows us to assume the validity of equation 5.10, this is not necessarily true in practice,, as we have not taken into consideration the effects of non-linearity and hysteresis in the other parts of the engine system, such as the injection and timing system.

The second point is that the value of K is unique to each given engine/dynamometer combination. Thus it would be necessary to perform a calibration or theoretical analysis everytime a change was made to the dynamometer or engine. It was felt that this would have the effect of cancelling out many of the advantages that this system of testing provides.

Because of the above drawbacks alternative approaches were considered.

5.2.2 - Quasi-Steady State Method

A different approach would be to consider that for some tests up to a given value of the speed acceleration (ramp rate), the difference due to the additional terms in equation 5.9 can be regarded as negligible when compared to the overall accuracy of the test. In other words this technique assumes that for low ramp rates, quasi-steady state conditions can be said to exist and that:-

$$Q_{E_{DYN}} \simeq Q_M \qquad (5.11)$$

This approach can be justified by a consideration of the practical conditions of engine testing and a closer examination of the relevant terms in equation 5.9. Testing of engines in the full load power curve mode is usually done in the normal operating range of the engine speed, and in this region the torque characteristic forms a fairly smooth curve. If our ramp rate is small then we can expect the values of all the terms on the right hand side of equation 5.9 to be small, with the exception of the Q_M term. Now in practicable terms the accuracy with which we can measure any parameter in our test system is limited, and as long as the errors introduced by the dynamic effect are small compared to the total system inaccuracies, their effect will not be significant.

Obviously, as the ramp rate is increased the significance of dynamic effect errors will increase, and one of the primary tasks of the investigation described subsequently, was to obtain data on the relationship between ramp rate and dynamic effect errors to enable one to define the maximum ramp rate compatible with a given level of error.

5.2.3 - Dual Ramp Averaging Method

A third alternative approach has its origin in the concept that the engine may be ramped in either a positive or negative manner. If the ramp has a positive gradient, torque generated by the engine will be expended in the acceleration of the engine and dynamometer and hence the measured torque will be less than the developed torque. For the case of a negative gradient, the opposite would be true, and the results will show a higher torque figure than the steady state value.

This method of obtaining a valid result for the developed torque, postulates that the torque differential is of equal magnitude at any given speed but of opposite sign for positive and negative ramps of the same magnitude. This statement leads us to a new derivation of the developed torque, as the average of the measured values obtained from two ramps of equal and opposite gradients. We can express our new approach as follows:-

$$Q_E > Q_M \quad \text{for} \quad \dot{\theta}_{ED} > 0 \tag{5.12}$$

and

$$Q_E < Q_M \quad \text{for} \quad \dot{\theta}_{ED} < 0 \tag{5.13}$$

Our postulate is given by: -

$$Q_{E_{\dot{\theta}_{ED}=k_1}} = 0.5 \left(Q_{M_{\dot{\theta}_{ED}=k_1, \ddot{\theta}_{ED}=+k_2}} + Q_{M_{\dot{\theta}_{ED}=k_1, \ddot{\theta}_{ED}=-k_2}} \right) \tag{5.14}$$

where $Q_{E_{\dot{\theta}_{ED}=k_1}}$ is the engine developed torque for an engine/ dynamometer speed of k_1.

$Q_{M_{\dot{\theta}_{ED}=k_1, \ddot{\theta}_{ED}=+k_2}}$ is the measured torque for an engine/ dynamometer speed of k_1 and a ramp gradient of k_2.

and $Q_{M_{\dot{\theta}_{ED}=k_1, \ddot{\theta}_{ED}=-k_2}}$ is the measured torque for an

engine/dynamometer speed of k_1 and a ramp gradient of $-k_2$.

It should be noted that this technique owes nothing in its derivation to the constant offset approximation mentioned earlier. We have not constrained our offset to remaining constant for any change in speed, and on the contrary it may vary in any way as long as the positive and negative gradient ramp offsets are of the same magnitude for any given speed.

So far we have concentrated on the torque produced by the engine in considering the relationship between dynamic and steady state testing. However, there are many other data parameters, such as pressures and temperatures, which are of importance to us, and which may well exhibit some form or other of dynamic effect. Because of this we will expand the applicability of our postulate of equation 5.14 to cover any parameter P. Using the same subscript meanings as before our new universal postulate is: -

$$P_{E_{\dot{\theta}_{ED}=k_1}} = 0.5\left(P_{M_{\dot{\theta}_{ED}=k_1,\ \ddot{\theta}_{ED}=k_2}} + P_{M_{\dot{\theta}_{ED}=k_1,\ \ddot{\theta}_{ED}=-k_2}}\right) \qquad (5.15)$$

It is this concept that the actual experimental work investigates in an attempt to prove or disprove it.

5.3 Preliminary Experimental Investigation

The first stage of the experimental program was accomplished by using one of the two test beds at Queen Mary College described in Chapter 3. The engine under test was a Ford 1300 c.c. OHV (overhead valve) spark ignition engine. The test stand equipment was not altered significantly from its standard form to allow the running of dynamic power curves. The only modification was the insertion of an analog filter to remove the effects of carcase jitter in the torque signal.

The first stage of the work was to develop a software package to allow the computer to ramp the engine speed demand whilst monitoring the speed and torque signals to provide data which would allow the alternative techniques described in the previous section to be re-examined on a practical rather than theoretical basis.

5.3.1 - Software Task Scheduling

The first major problem that we must consider in creating the required software package, is that of handling the scheduling of the various routines that go to make up the engine control programs. Routines involved in outputting set point demands, logging data parameters and performing man-machine interfacing are designated application programs whilst the programs concerned with the internal housekeeping of the processor, including task scheduling, are called system programs. Because the latter should be as efficient as possible in order to make their operation transparent as far as the overall

performance is concerned, they were all written in assembler language, in this case the CAP-16 (23) language of the General Automation SPC-16 computer that was to be used to handle the test stand. On the other hand this constraint did not apply to the applications programs which could be written in a high or low level language. However the high level language supported on the SPC-16 is Fortran IV (24), based on a rather inefficient single pass compiler with only limited optimisation facilities and so for the sake of general efficiency of execution and core utilisation CAP-16 assembler was used to as great an extent as possible.

Although we will look at the application programs in greater detail in a later section, we should perhaps first examine their nature and scope briefly in order to understand the problems that arise in writing the software for the system programs. The first stage is to list out the various tasks that the application programs must accomplish and this is done below.

i) Monitor engine speed to provide an alarm capability in the event of a control failure.

ii) Update the speed demand at regular intervals to generate a ramp whilst at the same time maintaining the correct engine throttle and temperature settings.

iii) Perform data acquisition for the specified engine parameters at the correct point in time.

iv) Print out the results obtained and plot the required graphs.

v) Initially bring the engine from idle condition to a full throttle condition with the engine speed at a suitable value

to begin the ramp.

vi) On completion of the test, return the engine to an idle condition and perform the necessary dialogue with the operator to determine the next action of the system.

vii) Receive from the operator the parameters required to define the specific nature of the test about to be run.

viii) Perform the calculation of derived parameters such as power from the raw data collected from the engine.

It can be seen that these tasks fall into two different categories, unique and multiple. In the case of unique tasks such as v, vi and vii above, the computer is only concerned with one action at a time and hence the task scheduling can easily be achieved on a sequential basis. The multiple tasks however require the interleaving time wise of two or more tasks to provide pseudo-simultaneous accomplishment of several tasks. To make matters worse the actual response of the various routines in real time can be quite critical. The alarm monitoring and the set-point updates have a very critical timing requirement as they must be executed at precisely the right moment. The data acquisition routines have a slightly less rigid timing requirement but nevertheless require execution within a fairly limited time slot to allow the data log points to be placed reasonably near their intended speed values.

Although it might be possible to perform the execution using sequential methods, the sheer complexity of the problem would make this very difficult and the overall structure would be very rigid

making it difficult to effect changes in the system behaviour such as altering the frequency of occurence of data acquisition. As a result of this it was decided that control of the ramp software should be based on a real-time executive technique to allow multi-programming.

The real-time executive programming allows flexible scheduling of many different programs which are listed in a table in the order of their priority. Each can be scheduled for execution at a particular point in time. However if during its execution a higher priority routine requires execution, processing of the first program is temporarily shelved until the higher priority routine has completed its execution.

5.3.2 - Special Real Time Executive

Most computer manufacturers offer one, if not several, real-time executive packages as standard software for their systems. Indeed General Automation have their RTX-16 package (25) for the SPC-16 and initially the possibility of using this as our controlling software was considered. However there are good reasons why the standard package did not fit the requirement. The problem arises from the fact that such packages are written with a view to providing the system software for conventional automated control schemes. In, for instance, the automation of a chemical plant, response times of several seconds are perfectly acceptable and hence it would only be necessary to run the executive program at comparatively long intervals. However in our application we are dealing with routines that may require response times in the millisecond range and if a package such as RTX-16 is used,

an inacceptably high proportion of execution time will be used in the housekeeping tasks involved in running the executive.

As a result it was decided that a special executive should be written that would be limited in scope to controlling the execution of only a few core resident programs. Such facilities as maintaining calendar information, controlling the input/output requests and allowing programs to be swapped in and out of bulk memory would not be included. This means that the executive itself can be simplified and hence run at shorter intervals without tying up a large proportion of the system execution time. However it is up to the application programs to ensure that there is no conflict in requests for input/output facilities.

Although for the testing described here we are only controlling one test bed and all the programs can be fitted into the SPC-16's 16K words of core memory, it would also be possible to run a multi-bed scheme by using a dual executive. In this case the overall system control would be handled by a comprehensive package such as RTX-16 which handled overall scheduling and allocation of core space and input/output requirements whilst a secondary, limited capability executive handled the detailed scheduling of fast response time programs.

The control software package which was written is called EXEC-16 and consists of several components. First there is an initialisation routine that is called to set up the system data areas. This routine also calls the necessary applications routine to initialise the test stand interface, take control of the engine and bring it to the starting point of the ramp. On completion of its tasks the routine act-

ivates the hardware for real time clock interrupts and hands over control to the real-time executive itself.

The real-time clock of the SPC-16 provides a high priority interrupt at intervals of 1000 instruction cycles. The processor cycle time of 960 nanoseconds thus results in interrupts occuring at 0.96 millisecond intervals. The interrupt terminates the current processing and vectors to a routine called RTCSR (for Real Time Clock Service Routine) whose algorithm is shown in Figure 5.3.

Its first task is to save the contents of the processor registers in a temporary buffer so that there is no loss of data when registers are used by the service routine itself. It then increments the basic time clock which is effectively a count of the number of clock interrupts that have occurred since the last time an executive scan was performed.

When configuring the software package the user sets up the executive response interval to define how often the program table is to be scanned. This value can also be altered at any time between tests without affecting any other part of the system timing, as the executive initialisation routine and other utility routines (see below) handle all the necessary calculations to ensure that the user applications routines requests for real time action in terms of seconds and milliseconds are translated into the system internal time scaling structure. In practice the package was run constantly with an executive scan interval of 9.6 milliseconds (10 interrupt cycles) although tests were made with other values to test the validity of the system timing programs.

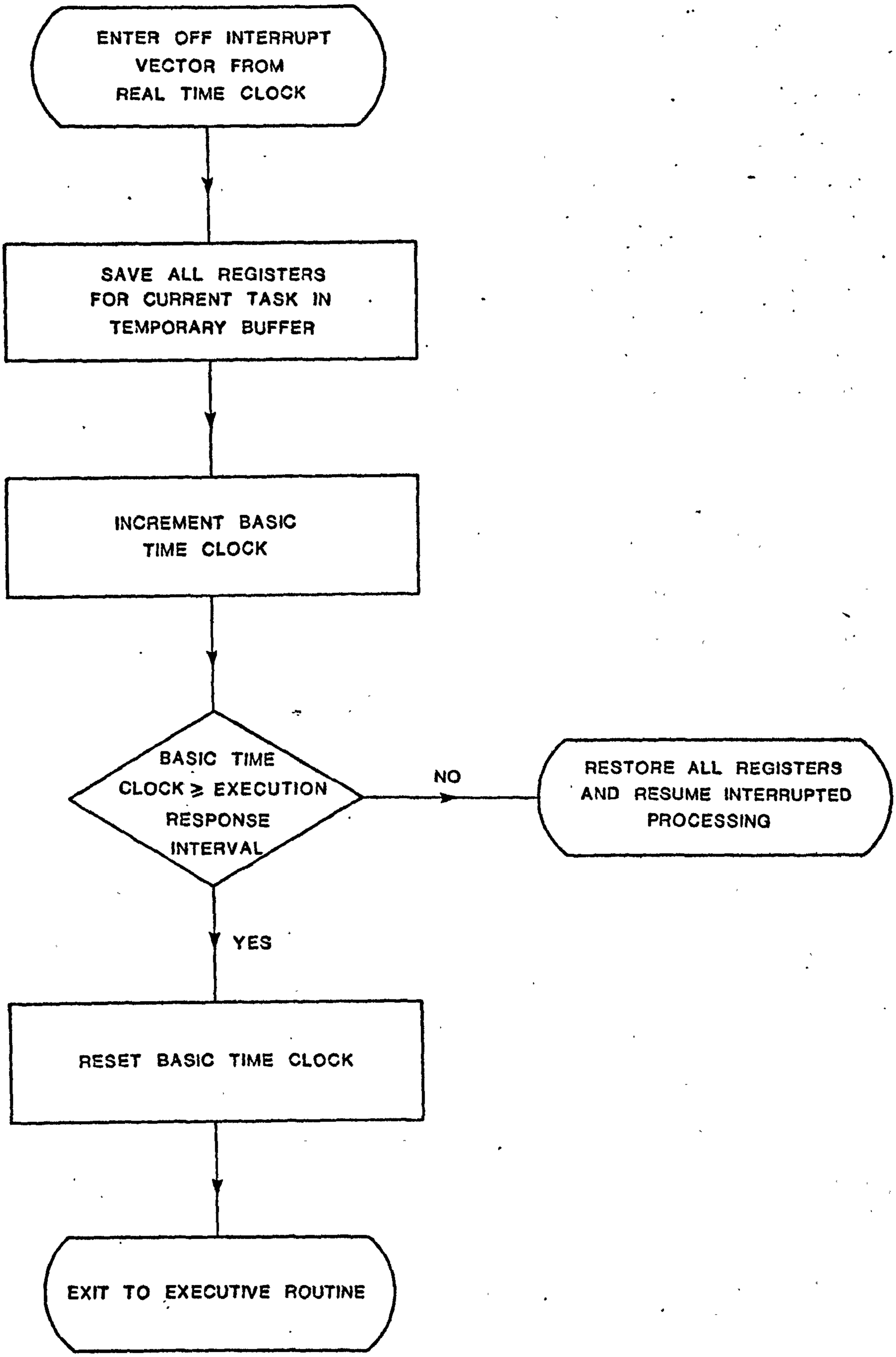


FIGURE 5.3 - EXEC-16 RTCSR ALGORITHM

After it has updated the basic time clock, RTCSR checks to see if another executive scan is due. If it is not the registers are restored to their condition on entry and processing of the interrupted program continues until the next real-time clock interrupt.

If a scan is due the basic system clock is reset ready to begin counting the next interval and control is passed to the executive routine itself.

The operation of the executive itself is based on the fairly common technique of a priority orientated program table (12). Each of the applications programs has an entry in the table consisting of 4 words as shown in Figure 5.4. The programs should be sorted into order of priority with the highest priority program appearing first in the table.

The use of the program table can perhaps best be illustrated by describing the execution of the scan routine PRSET whose algorithm is shown in Figure 5.5.

The first action of the scan routine is to update the system time clock. This consists of two words in memory STC1 and STC2. STC2 contains a count of the total number of clock interrupts, and is rolled over into STC1 by software when it reaches a value of 31,250. This means that each unit of STC1 is equal to 30 seconds in real time.

Next we are ready to begin a scan of the program table. Starting with the highest priority program, the setting of various bits in the right byte of the first word of the program entry is checked. If the first bit of the program status word is set, this indicates that the relevant applications routine is disabled and as there

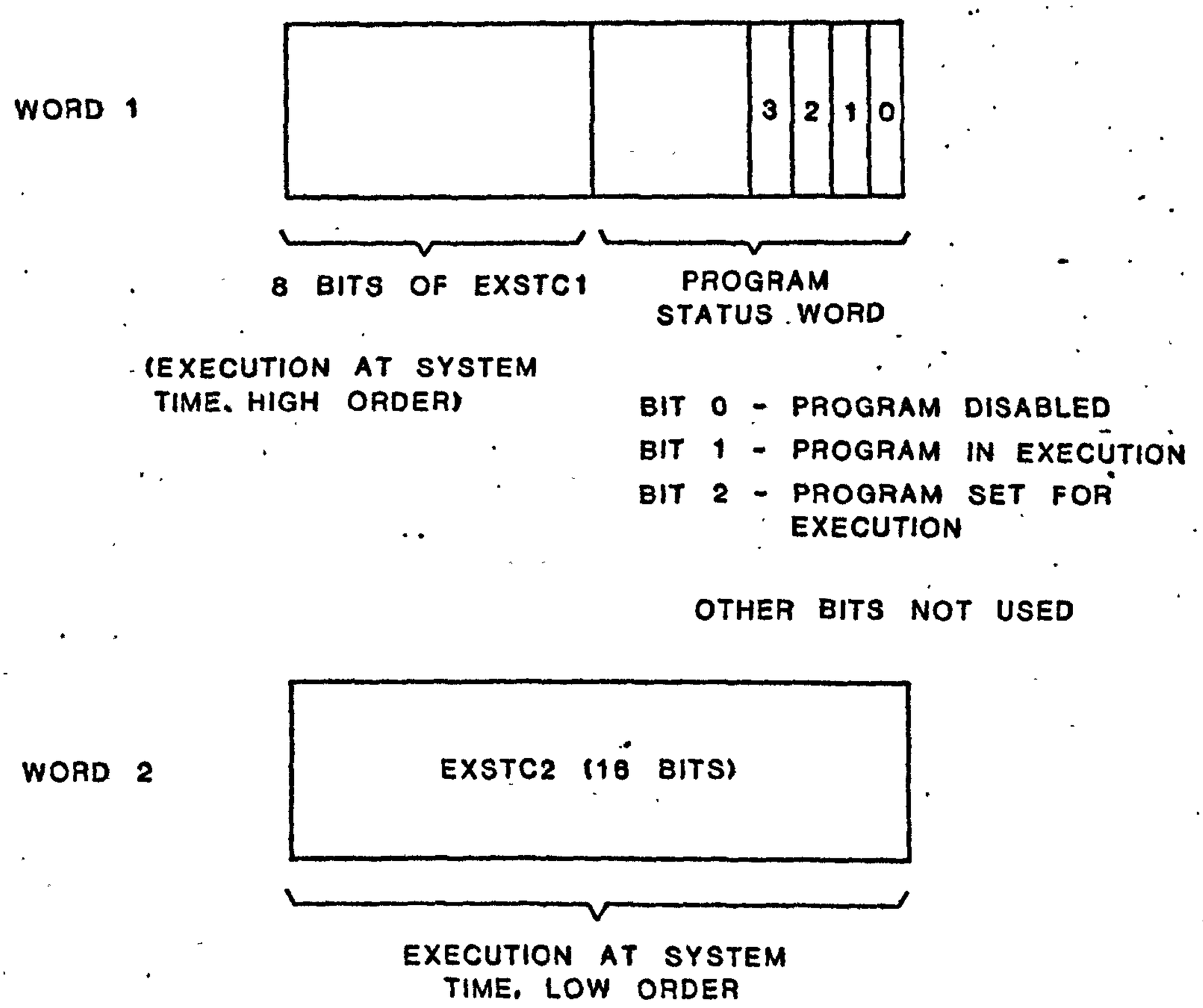


FIGURE 5.4 - EXEC-16 PROGRAM TABLE ORGANISATION

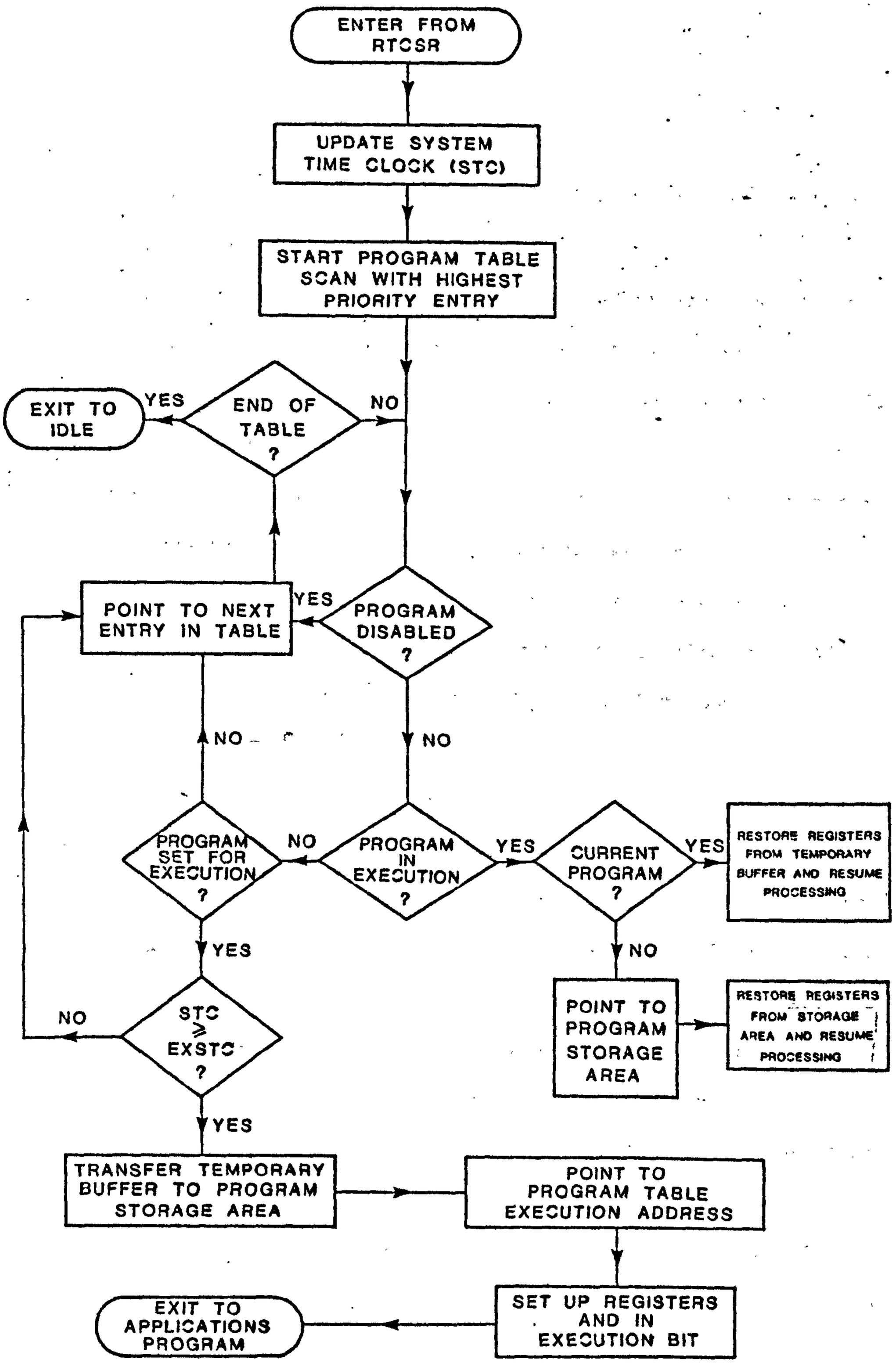


FIGURE 5.5 - EXEC 16 PRSET ALGORITHM

is no requirement for its execution the executive can proceed to the next entry in its program table.

If the status word does not show a disabled program the executive checks the program in execution bit. If this is set it means that the program is part way through its execution and was interrupted. If a check shows that in fact this is the program that was interrupted on the entry to RTCSR then all that is necessary is to restore the registers from the temporary buffer and resume processing.

The other alternative is that the program was interrupted at an earlier stage and another higher priority program has since been executed. Now in this case the program registers will no longer be in the temporary buffer. This is where the fourth word in the program table entry, SAVE, comes in. This is a pointer to an area of system memory reserved for the storage of registers whilst the program is not running. Once the registers are reloaded from this area, processing can be resumed as though the return was being made directly from a subroutine.

Another important facet of multi-programming is the protection of program data areas whilst its execution is temporarily halted. Many executives assign memory areas uniquely to each applications routine, a method that can be very expensive in terms of memory storage requirements. However as there is no mechanism to run a lower priority program without completion of all higher priority executions, under EXEC 16 it is possible to use a single dynamic stack technique to allocate run time data areas. By using the standard General Automation subroutine entry sequence for all applications routines, it is possible to ensure valid handling of data area addressing simply by

reloading an interrupted routine registers.

If a program is not shown as being in execution the last check the executive makes before returning to consider the next routine in the program table, is to check whether the program is set for execution. This is the basic mechanism for allowing the real time scheduling of the execution of the various applications routines. The left hand byte of word 1 in the program table and the second word, provide a time for execution value compatible with the system clock. The requirement is for the executive to start execution of an applications routine when the system time clock has a value equal to the EXSTC1 and EXSTC2 entries in its program table entry. If this has not yet come about, the routine is not yet due for execution and the scan can continue. Should the executive scan the whole program table without finding a routine at present requiring execution, then it passes control to an idling routine that holds the system inactive until the next real-time clock interrupt occurs.

When a program becomes due for execution, there are several operations the executive must perform. It may well be that to execute this routine it is necessary to shelve temporarily the execution of some lower priority program. In order to allow resumption of this routine later on, the registers saved in the temporary buffer must be transferred to that program's storage area shown by its SAVE entry in the program table. After this the executive is free to set up the new routine for execution. The address where execution is to be started is loaded from the EXADR entry in the program table and the registers set up to allow correct addressing of the next area in

the dynamic data storage stack. The executive must also alter the program status word to show that is in execution rather than due for execution. After this, control can be turned over to the applications routine.

In addition to the initialisation routine and executive, EXEC16 also comprises a set of utility routines that can be called by the user's applications routines as standard subroutines, to allow them to interface with the executive, in order to control the status and timing of themselves and other applications routines in the package. There are four such routines described briefly below:-

i) X$CMP - Execution complete. This routine is called at the end of an applications routine to terminate its execution. It would thus take the place of a Fortran STOP statement.

ii) EXSET - Execution set. In order to allow an applications routine to command the execution of itself or any other routine at a specific time relative to the present, this subroutine can be called. It has three arguments and the normal FORTRAN calling sequence would be: -

CALL EXSET (I, J, K)

returns here

where I = Number of program to be set (1 = highest priority program).

J = Incremental time of EXSTC1 (30 second units)

K = Remainder of incremental time in milliseconds (EXSET converts this into the equivalent number of real-time clock interrupts for setting EXSTC2).

When called, EXSET will set the EXSTC1 and EXSTC2 values of the referenced program to the current time plus the value given by J and K. It will also insert the set for execution flag in the relevant status word.

iii) LK$PR and ST$PR - Lock program and start program. This complementary pair of programs provide the ability to set or reset the program disabled flag of a program status word. They both have a single argument that is the number of the program in the table concerned.

5.3.3 - Applications Routines

We have seen how our high speed executive provides the capability to run a series of applications routines in a real-time environment. This is obviously a generalised piece of software that can be used in many different dynamic testing systems. However we now turn to the specific applications programs for the dynamic power curve. The programs described below are the original routines written for the preliminary investigation at Queen Mary College. Subsequent alteration and improvements are discussed later on.

The applications tasks can be divided into three broad categories:

i) Initialisation prior to performing the speed ramp

ii) Routines executed during the ramp

iii) Post - ramp routines.

Initialisation is performed by a single subroutine called SRAMP that is called by EXEC16 in its initialisation phase (see above). The actions of this subroutine can be seen clearly from its algorithm in Figure 5.6. Before attempting any take over of the test stand, the computer checks that the engine is in fact idling. For the engine under test the safe range was taken as 500 to 1100 rev/min. If the engine speed is outside this range the computer prints a message on the operators teletype, either SPEED TOO HIGH or SPEED TOO LOW, as applicable and enters a pause state ready to retry the test for idle condition.

When a valid idle condition is encountered, the computer sets its own digital to analog converter outputs for the speed and throttle position control to the idle values and then takes over from manual control to computer control in Mode III (direct throttle position setting and speed control by dynamometer loading). The program then gradually brings the engine to a full throttle condition, whilst at the same time setting the speed to the first of the logging points required for the test. This is accomplished by limiting the steps in throttle position to 20% and those for speed to 200 rev/min.

A further step that has to be taken is to reduce the engine speed by a further 150 rev/min. This is due to the fact that when we start to ramp the speed demand, there is an initial disturbance be-

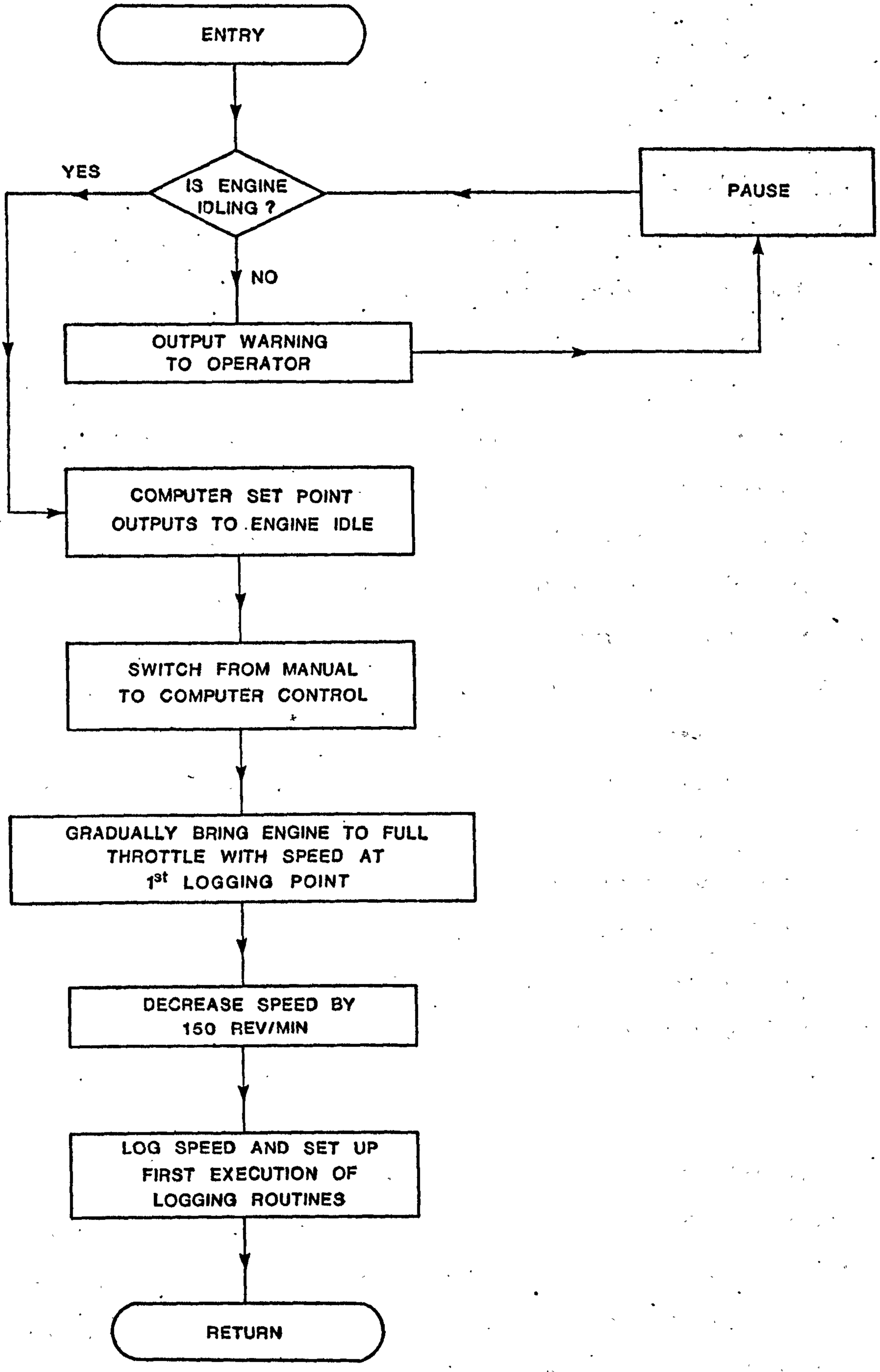


FIGURE 5.6 - SRAMP ALGORITHM

fore the engine speed settles on the ramp and if the engine is to be following our ramp smoothly when the first data logging point is reached then it is necessary to start the ramp early enough to allow sufficient settling time for the controllers. This initial non-linearity of the ramp can be seen in Figure 5.7.

Another factor that has to be taken into account is that we can seldom effect absolutely perfect calibration of our test stand. Thus for instance if we output a demand for 850 rev/min, we may find that we in fact get a somewhat different value from monitoring the speed. In steady state tests this can be overcome by providing an outer digital speed control loop that will modify the computer set point output. For our dynamic ramp this is unnecessary as it is sufficient that the engine sl ɔuld pass through its speed range in a ramp without having to bother about a controller offset, provided the data logging routines schedule themselves on the basis of the monitored data and are divorced from the demand output software. However as its final action SRAMP must log the current engine speed in order to predict accurately the occurrence in time of the engine speed reaching the first logging point so that the relevant logging routines can be activated at this point.

As can be seen from Figure 5.8 there are six mainline applications routines that form the EXEC16 program table in this software package. The highest priority program is a routine called UPDAT whose algorithm is shown in Figure 5.9. This program comprises three prime functions. These are the updating of the speed set point to generate a ramp, the check of the engine speed for alarm

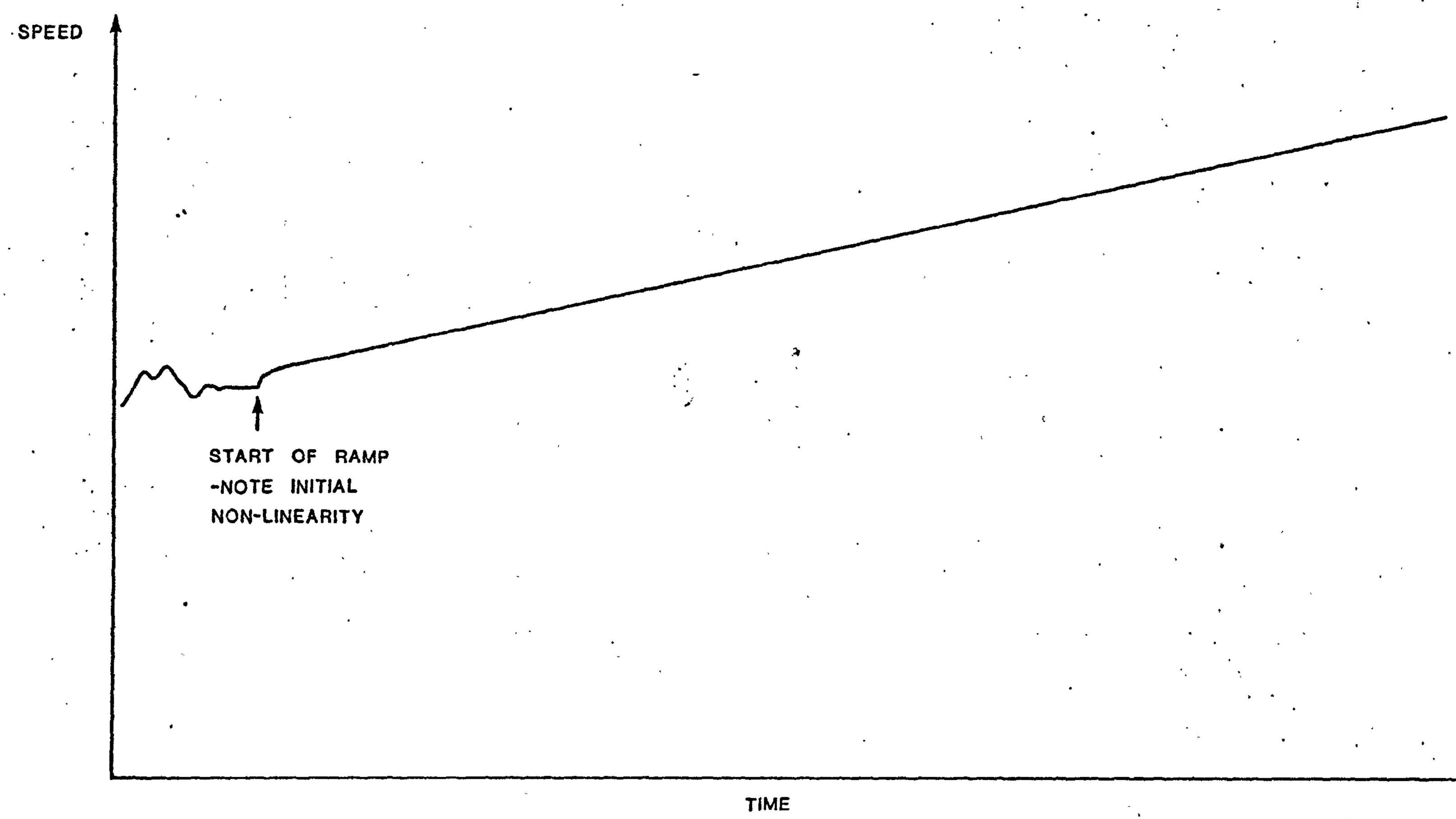


FIGURE 5.7 - SPEED AGAINST TIME (PLOTTED DIRECTLY)

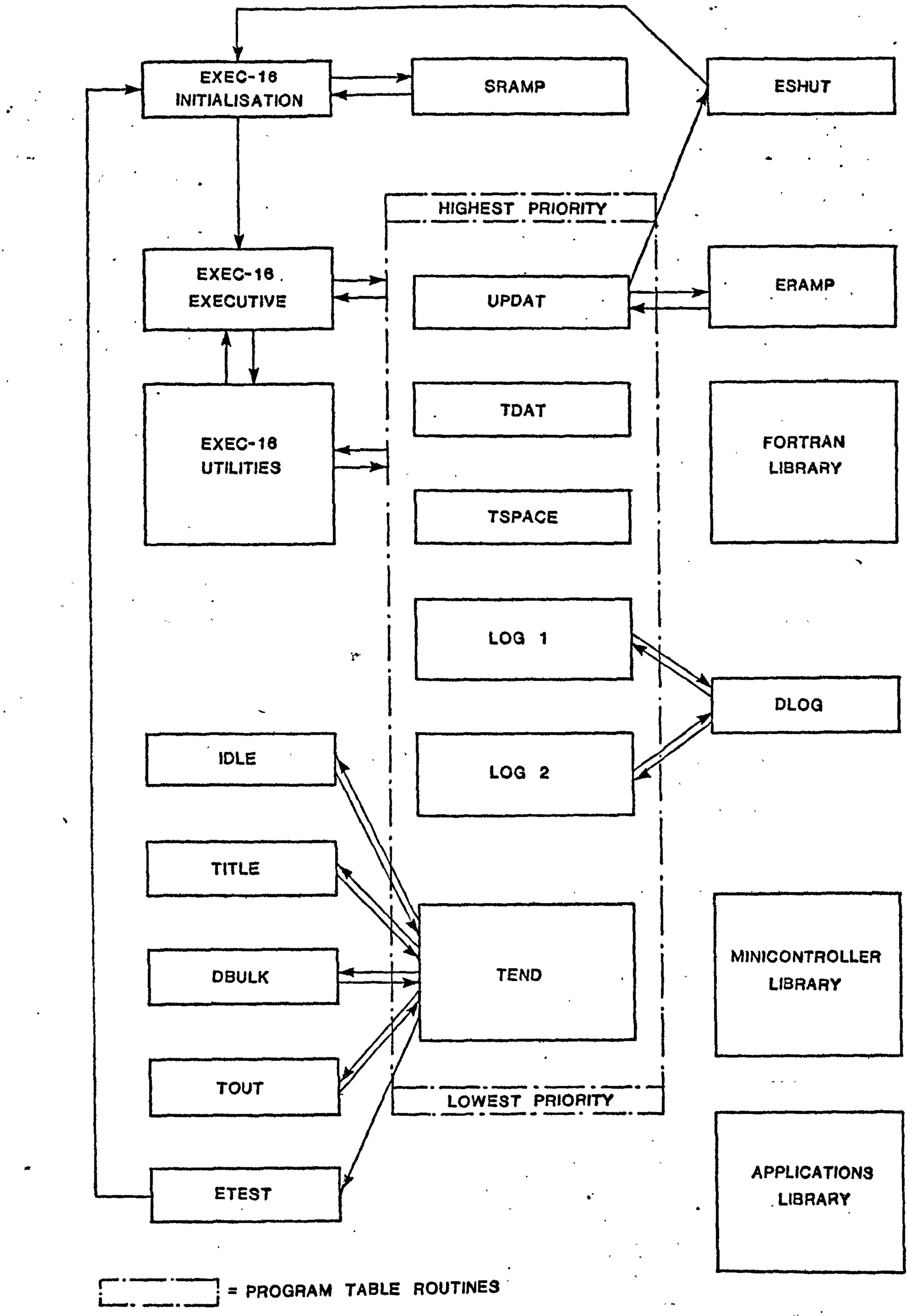


FIGURE 5.8 - DYNAMIC POWER CURVE PACKAGE

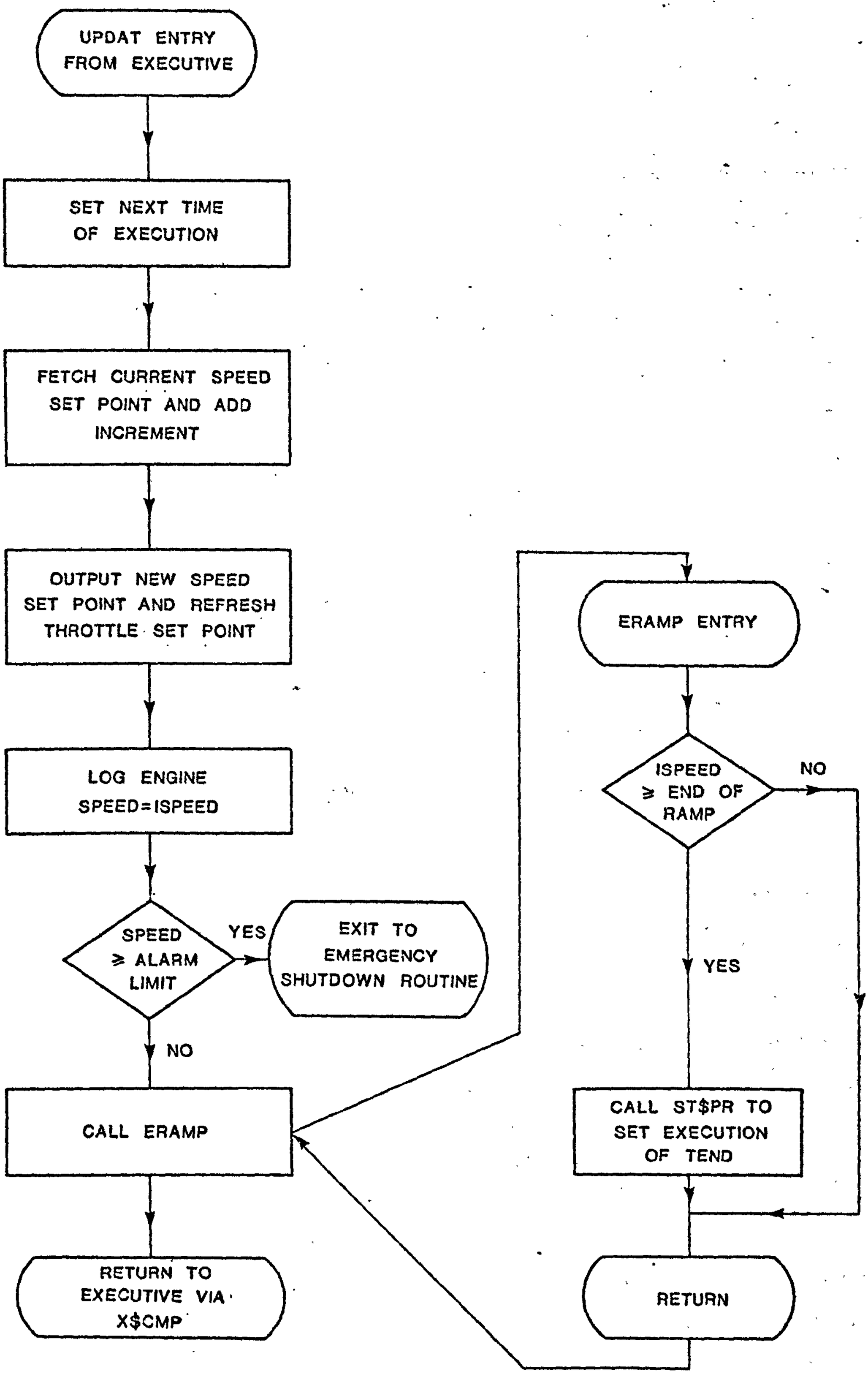


FIGURE 5.9 - HIGHEST PRIORITY ROUTINES UPDAT AND ERAMP

monitoring and the check to see if the end of the ramp has been reached. As all of these functions are of a high frequency and high priority nature it was thought expedient to combine them.

The first function is the updating of the speed set point. The current set point is accessed from a location in the system data area and a pre-calculated increment added to it. This incremental value is calculated during the initialisation phase and is given by the following equation: -

$$\Delta_{SPEED} = RAMP \times f_{UPDATE} \qquad (5.16)$$

where Δ_{SPEED} = Speed increment

$RAMP$ = Ramp Rate (System Acceleration)

and f_{UPDAT} = Frequency of execution of UPDAT

The values of the ramp rate and the frequency of execution of UPDAT can be set before commencing a test to allow evaluation of the system. The variation of the ramp rate is discussed in the following section. For this initial phase of testing, after some experimentation it was decided to execute UPDAT at 500 millisecond intervals as this was the largest value that gave a smooth speed ramp (see Figure 5.7) whilst a higher frequency would incur a penalty in terms of increased processor execution time.

The newly calculated value of speed set point is saved in memory and output to the test stand via the computer process control interface. At the same time the throttle position output is given

an update to ensure that the drift inherent in the digital to analog converters does not allow the throttle to move from its fully open position.

The next stage is for the computer to perform a data acquisition operation on the engine speed monitor channel. The value returned is used for both the alarm monitoring and the test for ramp completion. The alarm monitoring takes the form of a check against a single maximum permitted speed value. If at any time this value is exceeded then processing is passed to an emergency shutdown routine called ESHUT.

The first action of ESHUT is to attempt to shutdown the engine as quickly as possible. This is accomplished by transmitting a shutdown signal to the test stand hardwired logic and also by setting both the throttle position and speed set points to a minimum.

Unlike the majority of applications routines, ESHUT does not allow the computer interrupt system to become enabled when it starts. In this way, no further executive scans can occur so the program has effectively seized control of the system program sequencing. After commanding the test stand to shutdown, ESHUT disables the real-time clock interrupt mask bit thus aborting the test by disabling the EXEC16 executive. After the printing out a message to inform the operator that an emergency shutdown has occured, the whole test sequence can be restarted by returning to the beginning of the initialisation phase.

To allow an end of ramp condition to be detected UPDAT calls a subsidiary subroutine known as ERAMP. If this routine finds that the value of engine speed logged is greater than or equal to the value

stored as the last data logging point then it enables the execution of the end of ramp routine TEND that is the lowest priority program in the executive and is described below.

During initialisation the program table entry for UPDAT is set from immediate execution as soon as the EXEC16 executive becomes active.

i.e. $EXSTC1_{UPDAT} = EXSTC2_{UPDAT} = 0$

and PSW_{UPDAT} (Program Status Word) = 04_{16} (Set for execution).

After this, UPDAT is entirely responsible for scheduling itself. Immediately upon entry it reschedules itself by making a call to EXSET requesting another execution in 500 milliseconds time. This results in the routine's continued execution at regular intervals throughout the time the executive is operational.

The next four routines in the program table are all concerned with the business of logging data parameters from the test stand. The conventional steady-state power curve involves the logging of data at a series of points throughout the speed range of the engine under test. Usually, because of the time this takes, the intervals between logging points are quite large. However during the dynamic power curve the engine develops a continuous function of data parameters for the range and given the high speed data acquisition abilities of the modern minicomputer, the frequency of logging is only limited by the amount of memory available for data storage.

As a result of this situation it was decided to incorporate two separate data logging structures into the software package. The

first, defined as the major log, duplicates the conventional method by scanning and storing a number of data parameters at predefined points. The first and last of these points are also referred to by the SRAMP and ERAMP routines to define the boundaries of the required speed ramp. The data obtained is stored in a specifically assigned data buffer area and used as the basis of the printed results output on completion of the test.

The second set of data loggings comprise the minor log. This runs independently of the major log and occurs regularly at a predefined interval. In practice the available memory space limited this interval to a minimum of 20 rev/min for a ramp covering the normally full load power curve range of the engine under test. The data is stored in an open-ended buffer which begins at the highest memory location not used for other data or program storage and is subsequently used to plot the speed/torque/BMEP/power characteristics.

In actual fact both of the types of data log have two routines associated with them, a time sequencer and the actual logging routine. These are organised as follows: -

Highest Priority	1.	TDAT	-	Sequencer for minor log
	2.	TSPACE	-	Sequencer for major log
	3.	LOG 1	-	logger for minor log
Lowest Priority	4.	LOG 2	-	logger for major log

The reason for the adoption of this technique is to reduce the inaccuracies that may occur in the time scheduling if the execution

of a program should be delayed by the running of a higher priority routine. The sequencer routines are comparatively short and simple but the logging routines, with their need to access the process input/output interface, are not. Thus by using this divided technique, although a minor log may be under way (LOG1 is being executed), the scheduling of major log can take place because TSPACE has a higher priority even though the actual logging will not occur until the minor log is complete.

The algorithms of the scheduling programs are shown in Figure 5.10. It can be seen that there is a great deal of similarity between the two. Both routines start by scheduling their respective logging routines for immediate execution by making calls to EXSET with the relevant arguments.

In both cases the next task is the rescheduling of each routine by itself. The most important aspect of these actions is that there is a purely theoretical approach used. The prediction is based on the specified ramp rate rather than as a result of measurements of the actual engine performance. For the minor log the prediction is simplified by the fact that the interval between logs is always constant. As a result of this an incremental time value is calculated during the initialisation phase and the TDAT routine simply makes a call to EXSET to reschedule itself using the predefined value as an argument.

In the case of the major log, the points at which logging is to occur are stored as a series of engine speed values in memory. As the interval between subsequent loggings may not be constant,

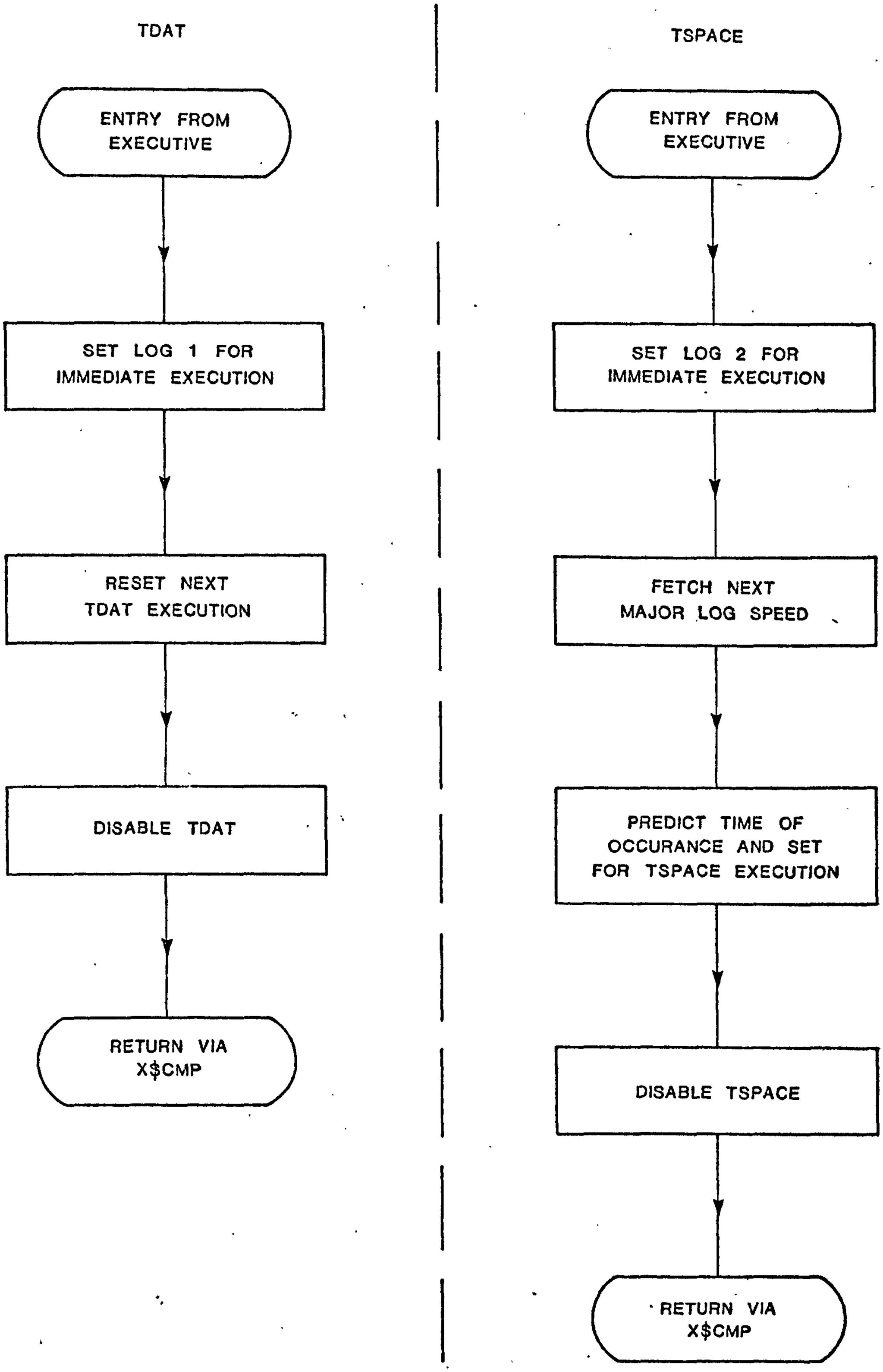


FIGURE 5.10 - TDAT AND TSPACE ALGORITHMS

TSPACE must access the table containing the logging points and perform a calculation to predict the point in time that the next logging will occur and then make the relevant call to EXSET to reschedule itself. The prediction is made on the basis of equation 5.17.

$$t_{n+1} = t_n + \frac{N_{n+1} - N_n}{dN/dt} \qquad (5.17)$$

where t_{n+1} = Time of next logging

t_n = Time of current logging

N_{n+1} = Engine Speed at next logging

N_n = Engine speed at current logging

$\frac{dN}{dt}$ = Rate of change of engine speed (ramp rate)

The final act of both TDAT and TSPACE is to lock themselves out of execution by way of a call to LK$PR. The prime reason for this is to avoid any risk of conflicting scheduling that could occur in the case of short logging intervals combined with a high ramp rate. This could possibly lead to an attempt to schedule the next logging before the current one has been completed. This lock out technique also provides an additional advantage in that the disablement of a program is the first condition checked by the executive and hence the overheads of processing time associated with the system executive are reduced.

If we now turn to the two logging routines themselves shown in Figure 5.11, it can be seen that for efficiency they both use a common subroutine to log the engine speed and torque. The algorithm of this routine, called DLOG, is shown in Figure 5.12. The program fetches a series of values of speed and torque from the minicomputer process interface with a delay between each reading and finally returns the average value on completion. The delay is effected by making a call to a subroutine called QSDLY that forms part of the Minicontroller Library (see Figure 5.8) which is a collection of utility routines primarily providing driver routines to handle input/output via the process interface. QSDLY is a software time delay routine called with an argument that represents the number of milliseconds for which execution is to be delayed.

Both the number of points averaged (NUM), and the delay between points (IDLY), can be varied but initially DLOG was set up to average five values at intervals of 10 milliseconds. The purpose of this was to remove any high frequency random noise from the signal whilst at the same time keeping the total span of the readings (50 milliseconds) sufficiently short so as not to contradict the dynamic nature of the test.

In the case of the minor log all that is necessary is to store the values of speed and torque returned by DLOG in the data buffer and having updated the pointer to the next available location in the buffer, remoye the lockout status of TDAT and conclude processing until recalled at the next scheduled logging point.

The major log routine however is equatable to the type of logging

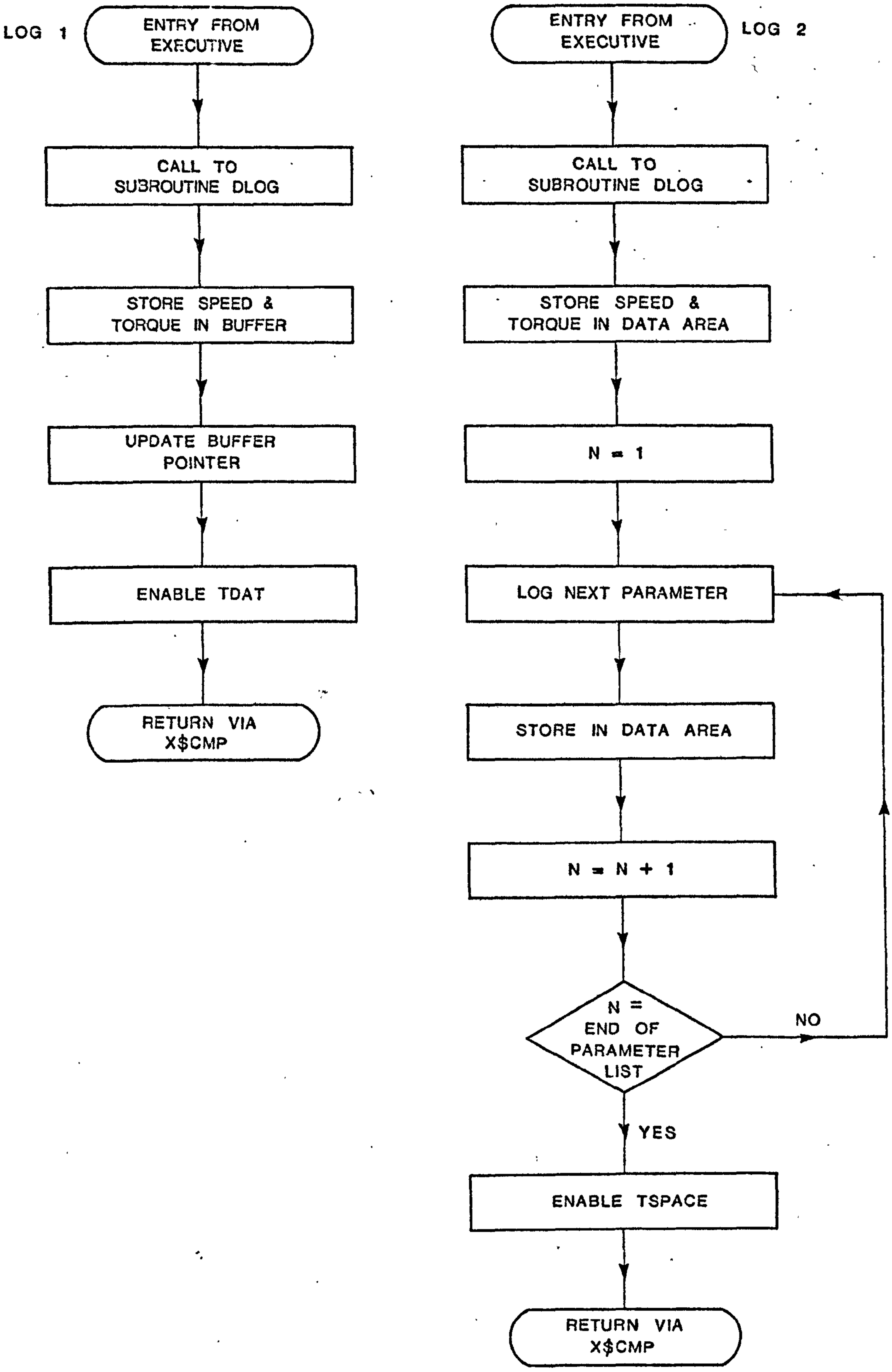


FIGURE 5.11 - LOG 1 AND LOG 2 ALGORITHMS

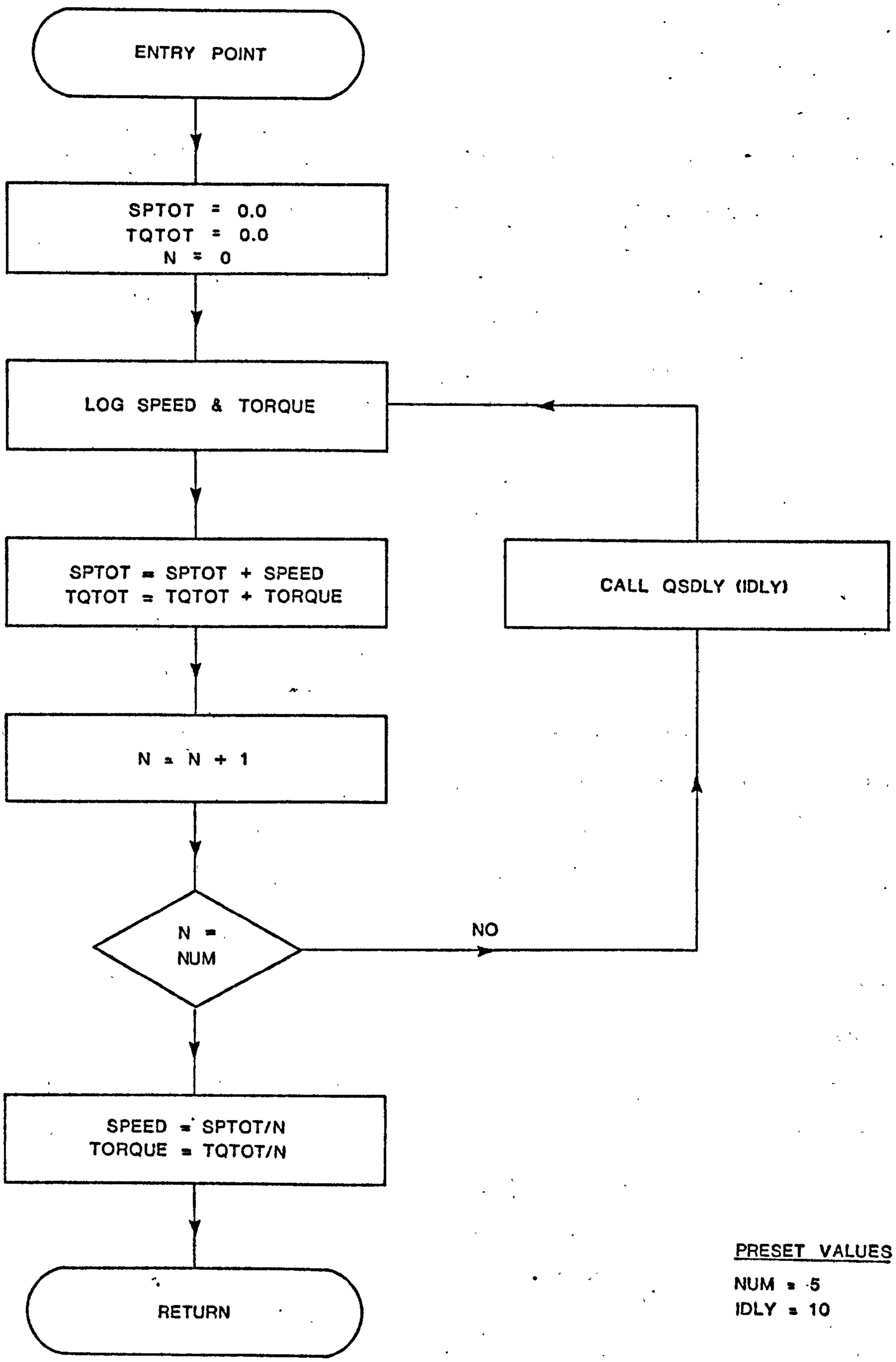


FIGURE 5.12 - DLOG ALGORITHM

performed during the steady state power curve and hence includes the ability to scan a range of other parameters taken from a predefined list. In the case of the preliminary testing these parameters were: -

1) Air Temperature (Ambient)
2) Cooling Water Inlet Temperature
3) Cooling Water Outlet Temperature
4) Oil Temperature
5) Exhaust Temperature
6) Oil Pressure

These are logged straight from the process interface by way of calls to the relevant input/output routines of the minicontroller library. The data obtained is then added to the torque and speed values stored in the assigned data area. Finally the routine enables TSPACE and terminates its execution.

The lowest priority routine TEND is set up for immediate execution but with its disable flag set during the initialisation phase. Thus during the normal running of the speed ramp it will remain inactive. However, as mentioned earlier, when the end of the ramp is reached, the subroutine ERAMP will remove the disabling flag. Any logging processes currently being operated or immediately due will first be completed as they have a higher priority. Eventually however the executive will pass control to TEND. Its first action (see Figure 5.13) is to disable the real time clock interrupt thus

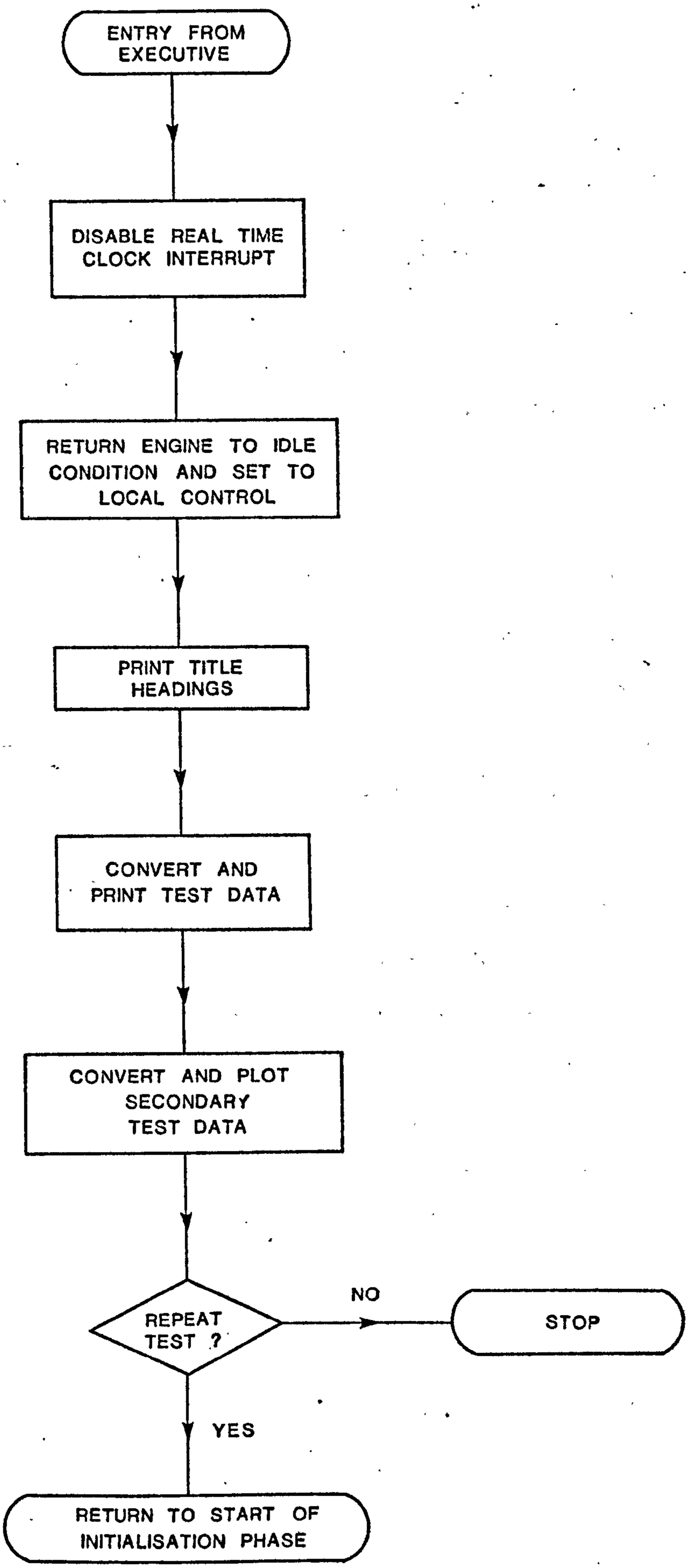


FIGURE 5.13 - TEND ALGORITHM

terminating operation of the EXEC16 executive. Because of this there will be no further updates of the speed set point and TEND will act as a straightforward sequentially executed program.

Before proceeding to the data output phase TEND gradually returns the engine to idle conditions (no load, throttle fully closed) and then disables the computer control to allow manual control of the test stand.

The first part of the data output phase is to print the data log headings on the teletype followed by the list of parameters comprising the major log for each logging point. All values stored in the data area are simply in terms of the voltage monitored at the process interface and must therefore be converted to the relevant engineering units.

It should be noted that we have stored for each point the speed and the actual measured torque. The subroutine DBULK that is responsible for printing the data also calculates the measured power as well. In addition the program follows the standard testing procedure of calculating a corrected torque figure to make allowance for variation in ambient temperature and pressure. The correction equation used is given below, and is a standard Ford correction equation.

$$\text{CORRECTED TORQUE} = \text{MEASURED TORQUE} \times 0.85\left[\sqrt{\left(\frac{273.33 + T_A}{288.88}\right)} \times \left(\frac{75.9968}{\text{BARP}} - 0.15\right)\right] \qquad (5.18)$$

where T_A = Ambient Temperature (^oC)

BARP = Barometric Pressure (cm Hg)

The current barometric pressure is entered manually during the initialisation phase and the logged value of air temperature at each stage is also used. On the basis of this the computer calculates and prints the corrected values of torque (load), power and brake mean effective pressure (BMEP). All these calculations are performed by a series of calls to the subroutines of the applications library (see Figure 5.8), a group of programs specially written for mathematical processing in engine testing.

The next part of the data output phase is the direct plotting of the engine performance characteristics. Three curves are plotted:

1) Engine Speed vs. Corrected Torque

2) Engine Speed vs. Corrected BMEP

3) Engine Speed vs. Corrected Power

The data used is the speed and torque values obtained from the minor loggings. The plotting subroutine TOUT used the same correction factor calculation routines as those used by the printing program. However a problem arises in that we do not have a value for ambient temperature at each point. It would of course be possible to include a log of this parameter in the minor log but this would increase the execution time and data storage required thus reducing the maximum frequency of minor logs. Instead it was decided that a figure for air temperature based on an interpolation between values obtained in

adjacent major logs should be used. As the rate of change of ambient temperature will be fairly low this does not introduce significant errors. Thus for a given speed point in the minor log the air temperature is calculated on the following basis:-

$$TA_N = TA_n + \frac{(TA_{n+1} - TA_n) \times (N_N - N_n)}{(N_{n+1} - N_n)} \qquad (5.19)$$

and $$N_{n+1} \geq N_N \geq N_n \qquad (5.20)$$

where parameters TA = Ambient Temperature

N = Engine Speed

and subscripts N = Minor log point

n = Preceeding major log point

n + 1 = Subsequent major log point

The barometric pressure used is again that entered by the operator during the initialisation phase.

The plotting routine draws a series of straight lines between intervening points. However if the frequency of minor logs is high then the plots will appear to be the continuous curve of the engine characteristics.

The final action of TEND is to transfer to a termination routine which allows the operator to state whether he has finished or whether he requires another test run, in which case control is passed back to the beginning of the initialisation routine.

5.3.4 - Preliminary Results

After a few trials to prove out the basic software package, a series of tests were run to provide experimental data on the overall feasibility of running a computer controlled dynamic power curve and to allow a preliminary examination of the relationship between results obtained in steady state and dynamic conditions.

The tests took the form of a series of power curves run at different ramp rates. The values of system acceleration used were varied between 100 rev/min/min (0.175 rad/s^2) and 4000 rev/min/min (6.98 rad/s^2).

Several runs were performed at each ramp rate to allow assessment of the repeatability of the results. The corrected values of torque showed a high level of repeatability.

In an attempt to examine the validity of our approximation of considering the dynamic power curve as 'quasi-steady state' (see Section 5.2.2) a series of results was used to compare increasing ramp rates with a baseline. As at this stage the software package did not include the direct capability of running steady state power curves, it was decided to identify a very low ramp rate as our baseline. The value chosen for this was 200 rev/min/min (0.35 rad/s^2) and the reason for picking this ramp rate was as follows. A series of tests showed that there was no significant difference in torque output between the 200 rev/min/min ramps and even slower ramps and results from steady state tests within the scope of the accurracy of the measuring system.

The result of a series of evaluation tests for different ramp rates are shown in Figures 5.14 to 5.18. The deviation from our baseline as a function of the ramp rate is shown in Figure 5.19. With ramp rates of 600 rev/min/min (1.05 rad/s^2) and below the deviation is too small to be plotted accurately.

If we consider our approximation of equation 5.11 it can be seen that Figure 5.19 can be used to set our maximum ramp rate which validates the equation to within a given degree of accurracy. Thus, for example, if our required accuracy was 1% it would be perfectly permissible to use a ramp rate of 1000 rev/min/min (1.75 rad/s^2).

During the course of the preliminary testing, a problem was encountered in the accuracy of positioning of major logging points. Several tests would be run with the loggings being obtained within ± 5 rev/min of the desired speed value, but every so often a 'rogue' would occur with all its points out by as much as ±50 rev/min. The cause of this malfunction proved difficult to pinpoint. In a series of tests run sequentially using exactly the same parameters and conditions the rogue would occur interspersed with a whole run of perfect tests. It became clear that the cause of the problem lay in the system hardware rather than the software with the functioning of the digital to analog converter system being a prime suspect.

The solution adopted was typical of the approach often used in modern automation, i.e. that of using computer software processing abilities to alleviate the shortcomings of the system hardware. The major log scheduling routine was modified to include an element of interactive optimisation with the test stand as shown in the new

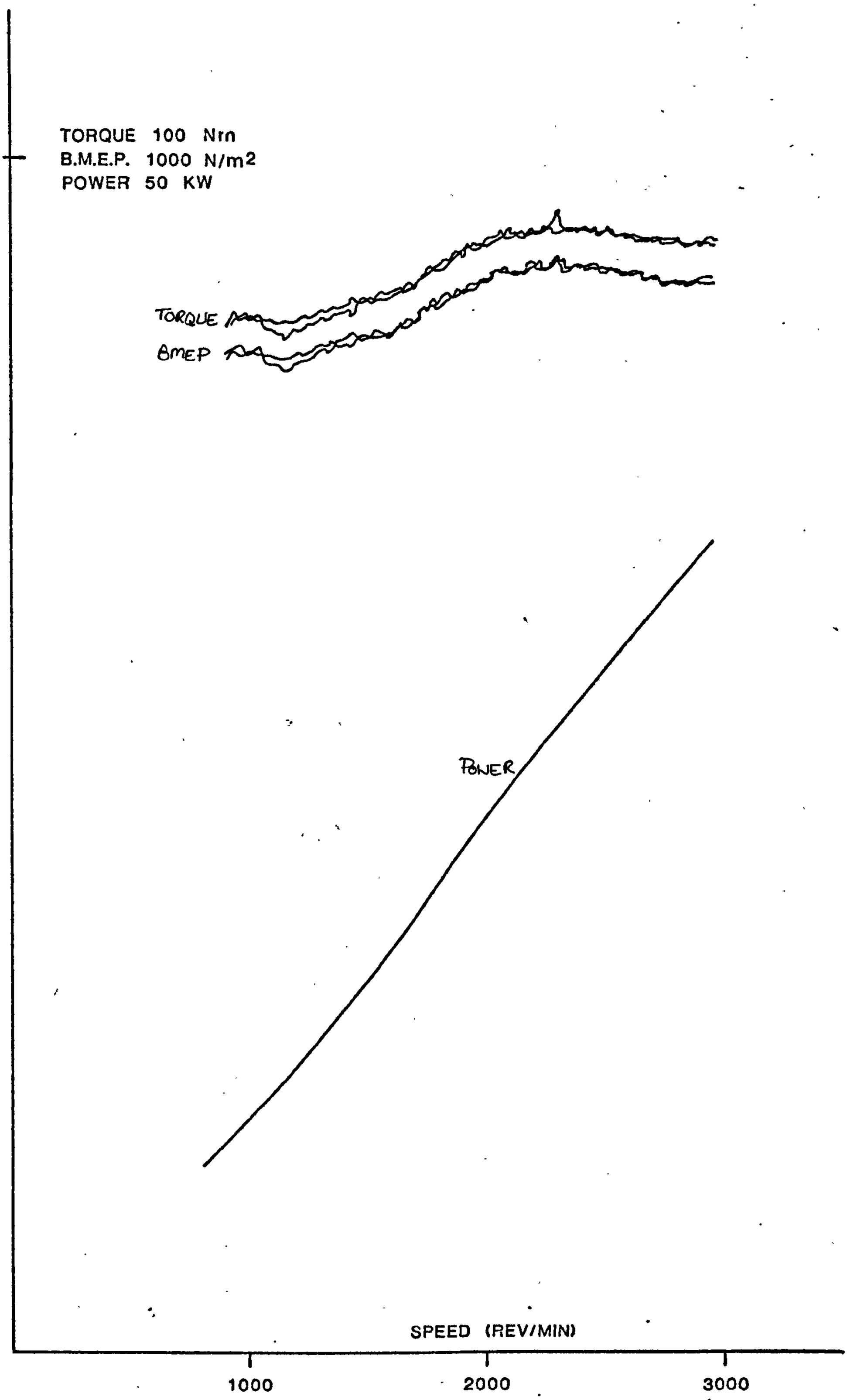


FIGURE 5.14 - SPEED TORQUE CURVES AT 200 AND 600 REV/MIN/MIN

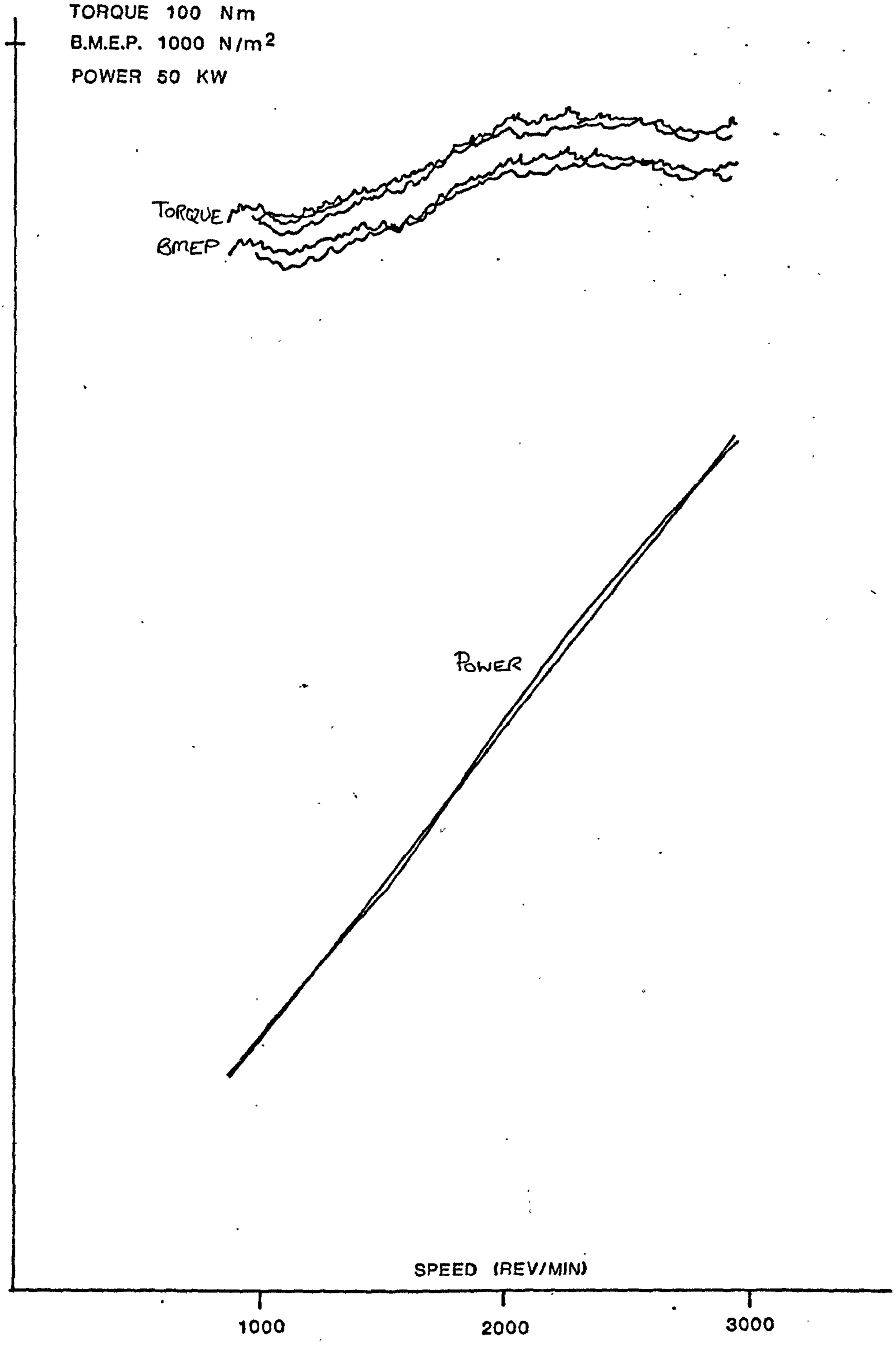


FIGURE 5.15 - SPEED TORQUE CURVES AT 200 AND 1000 REV/MIN/MIN

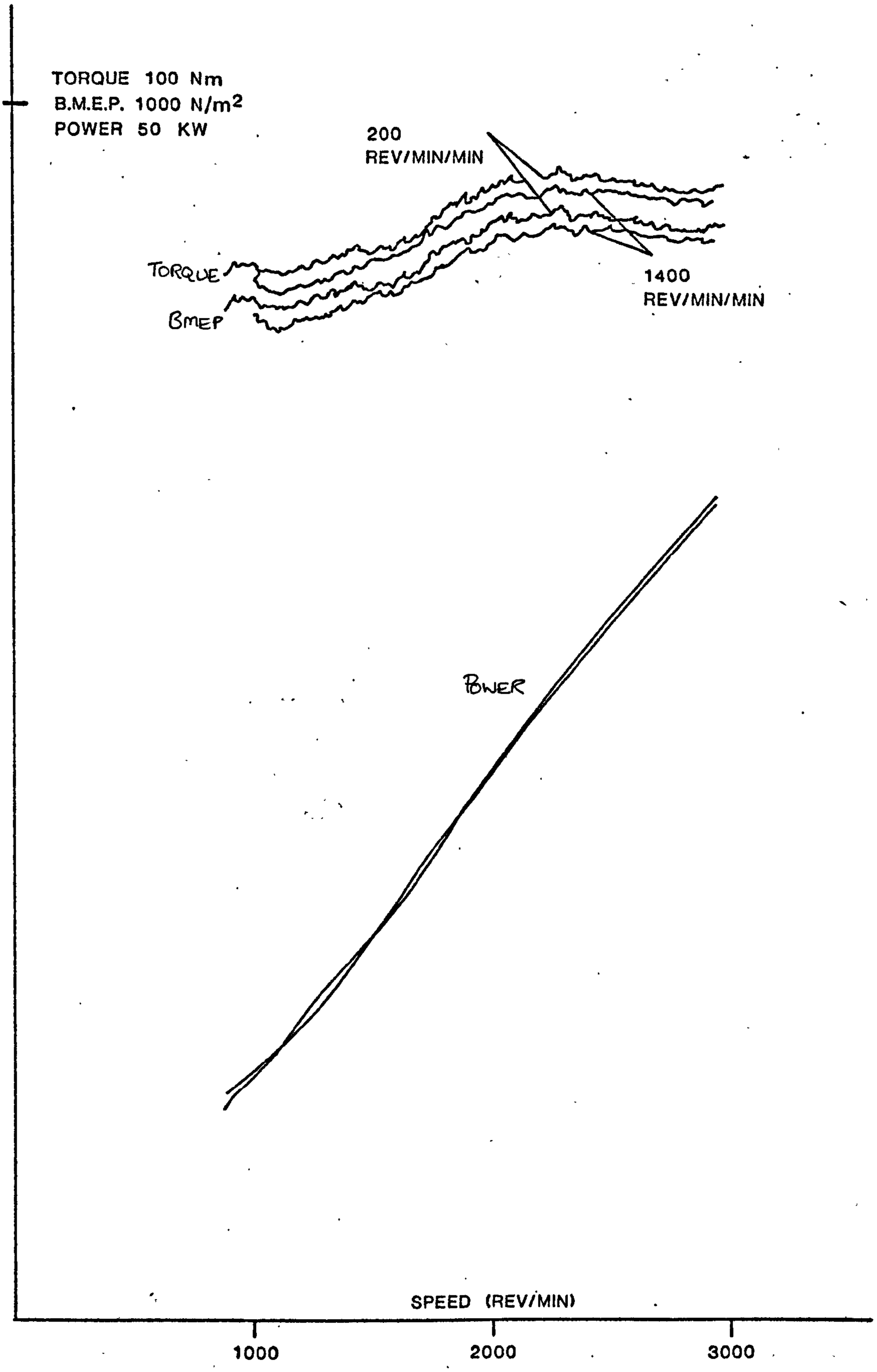


FIGURE 5.16 - SPEED TORQUE CURVE AT
200 AND 1400 REV/MIN/MIN

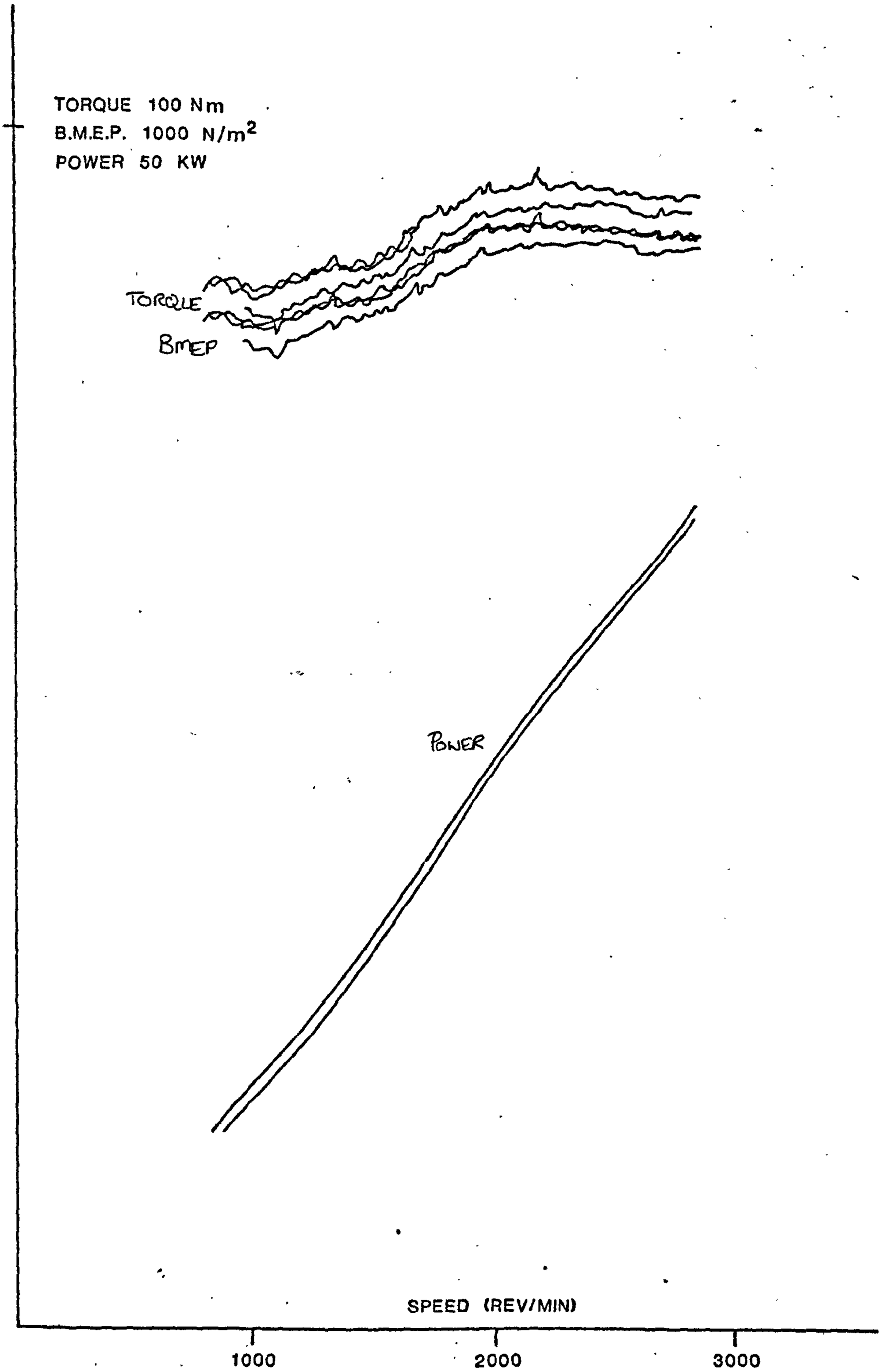


FIGURE 5.17 - SPEED TORQUE CURVE AT
200 AND 1800 REV/MIN/MIN

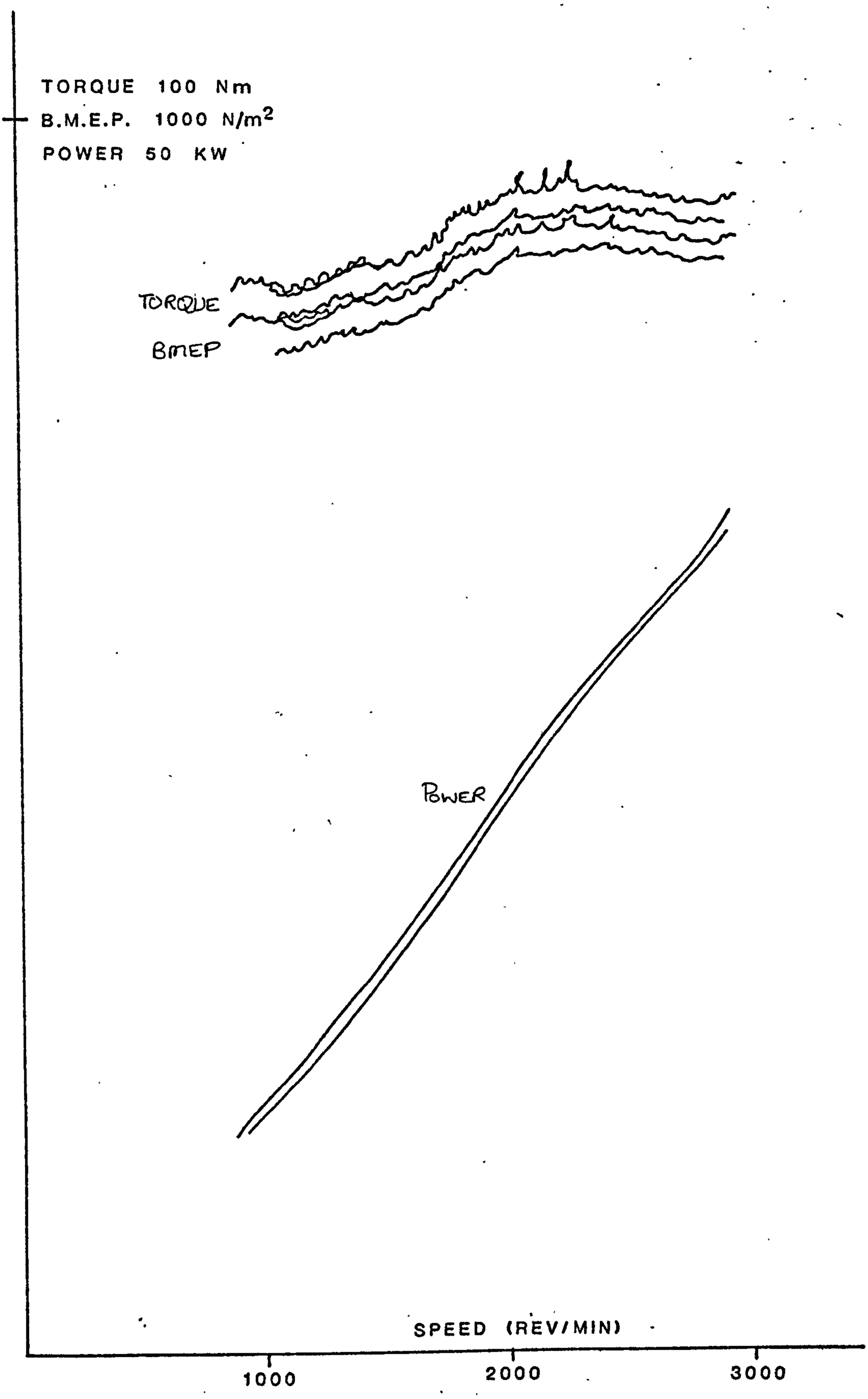


FIGURE 5.18 - SPEED TORQUE CURVE
AT 200 AND 2000 REV/MIN/MIN

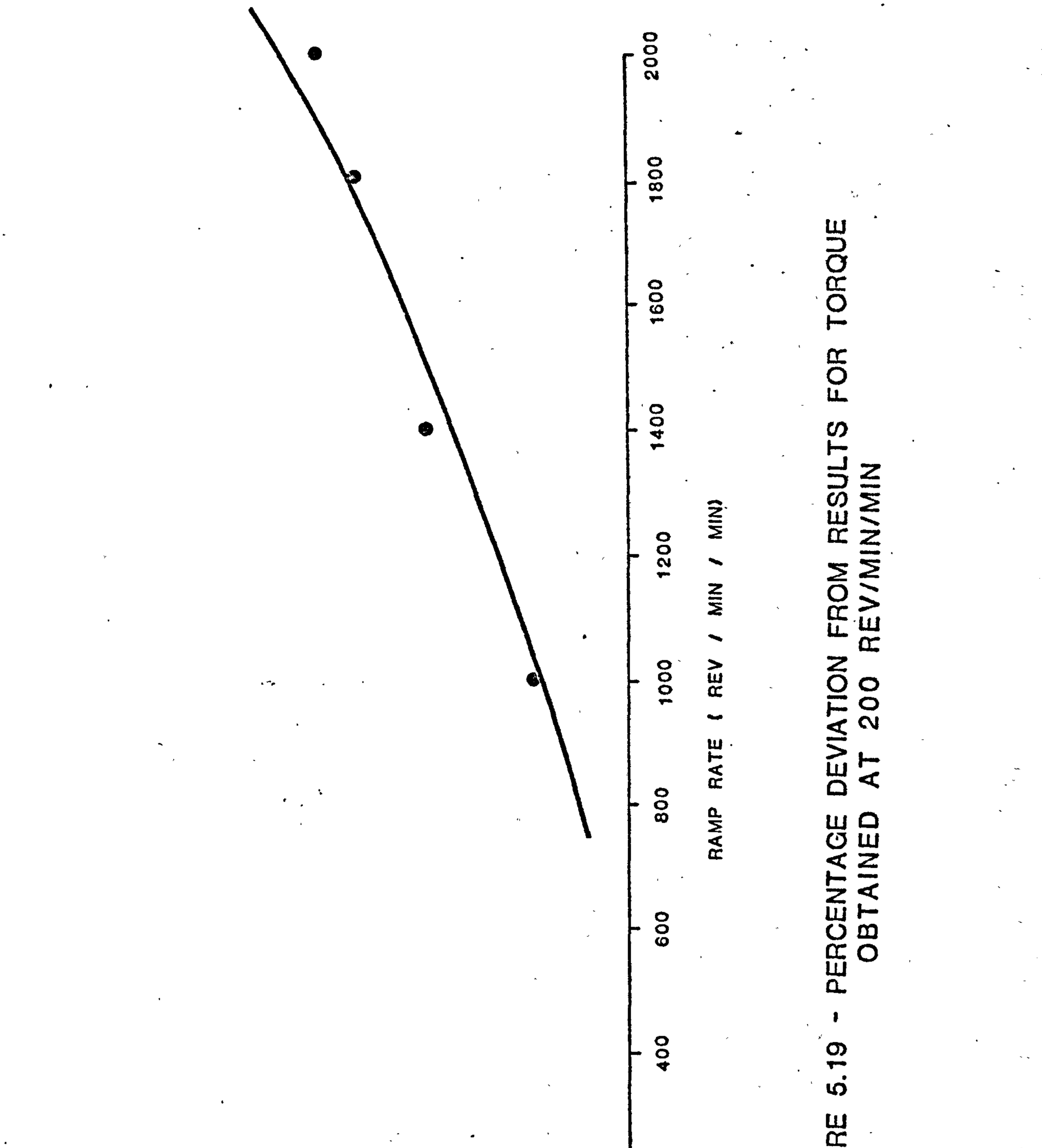


FIGURE 5.19 - PERCENTAGE DEVIATION FROM RESULTS FOR TORQUE OBTAINED AT 200 REV/MIN/MIN

algorithm of Figure 5.20.

The new version of the routine adjusts the scheduling to allow for any variation from the ramp. This is accomplished by having the first execution of TSPACE scheduled for a time predicted as being the point 50 rev/min before the next logging point. TSPACE then enters a time delay loop by checking the speed and resetting itself for execution after a short delay until the desired speed is reached. Only then is the actual logging routine scheduled. Using this method very accurate synchronization can be achieved. The scheduling of the higher frequency minor log points does not require this accuracy and hence remained unaltered.

5.4 Power Curve Evaluation

The preliminary part of the investigation showed the basic feasibility of using a computer controlled test stand to run dynamic power curve tests on an internal combustion engine. In addition the data obtained showed the increasing hysteresis effect for unidirectional ramps. These results were sufficiently encouraging to make it important to perform a more detailed investigation into the behaviour of the dynamic power curve and its feasibility as a replacement for conventional testing techniques.

The objectives of the new investigation were to be as follows:

i) That other types of engines should be tested using the dynamic method to ensure that universal applicability could be assumed for the findings.

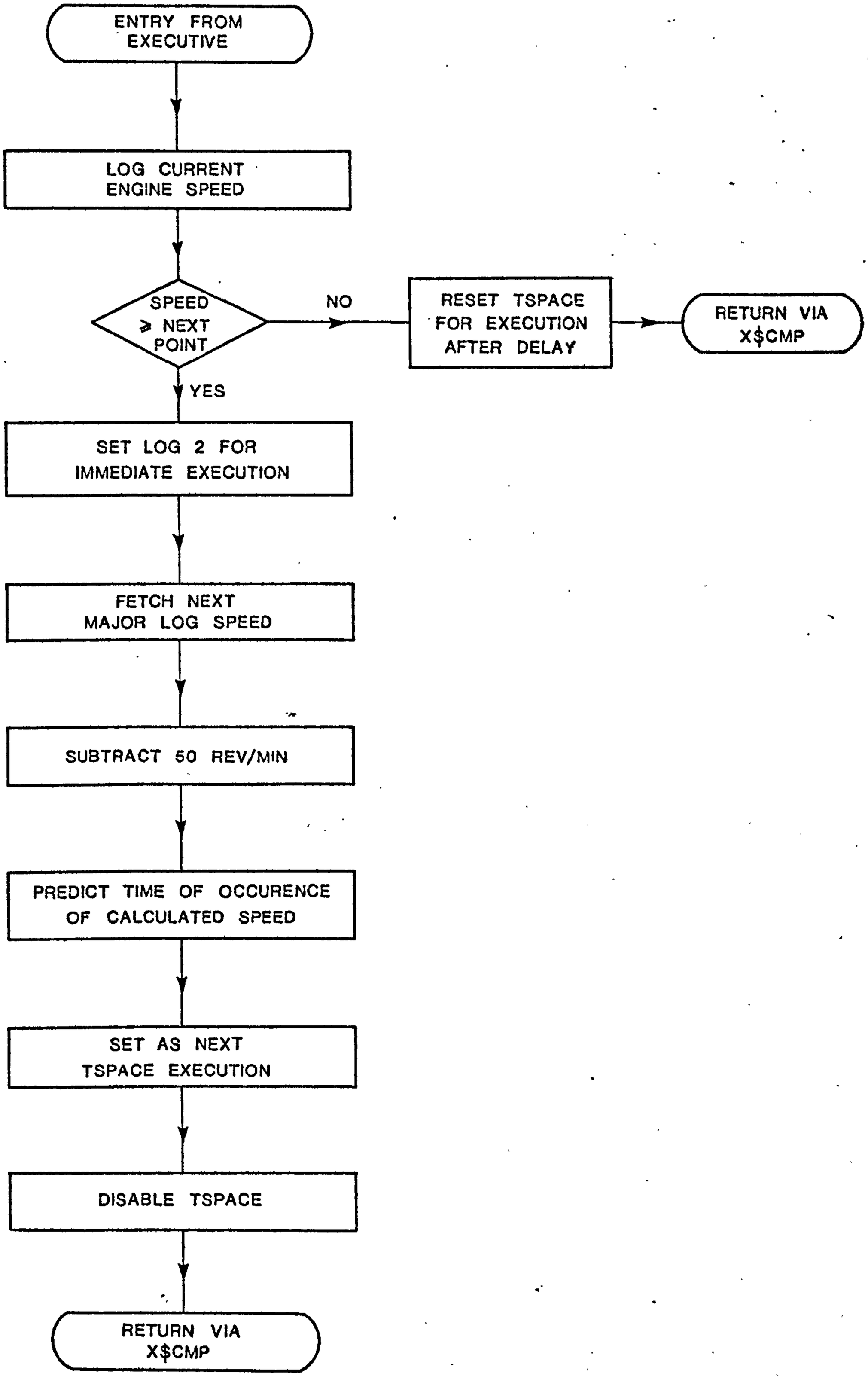


FIGURE 5.20 - MODIFIED TSPACE ALGORITHM

ii) The software package should be modified to include all the additional facilities found in a practical, industrial type package such as alarm condition monitoring and full operator/machine dialogue interface so that the tests would include a realistic approach to the system task load.

iii) Dual ramp ability should be included to allow both the single acceleration ramp of the preliminary package, as well as a dual ramp test consisting of an acceleration ramp followed by a deceleration ramp. This would allow the dual ramp averaging method of section 5.2.3 to be evaluated.

iv) The package should also include the ability to perform steady-state power curves so that the two types could be run side by side to allow direct comparison.

v) The data logging, printing and plotting functions should be increased in scope to allow the effect of dynamic conditions on other data parameters to be investigated.

5.4.1 - Software Package Modification

It was planned that the actual testing should take place using a test cell made available at the Ford Motor Company Dunton Research and Engineering Centre. The test cell to be used was virtually identical to those belonging to the Ford six cell system described in Chapter 3, except that there was no connection to the computer. Instead, the cell was connected to the General Automation SPC16 (used in the preliminary tests) which was moved to Dunton for the purpose. As the six cells, and hence the additional cell used, were based on

the test beds built at Queen Mary College, there was little special interfacing required but a small interface unit was designed and built to match electrically incompatible signals.

In fact much of the work on rebuilding the software package was done at Queen Mary College and the functioning of the new system was checked and de-bugged using one of the analog computer test stand simulations described in Chapter 7.

The new software package was made up by using modified versions of two already existing packages, these being the previously described dynamic power curve package and the steady state procedure described in Chapter 4. Both were modified so that they became subroutines to a new master program which also included improved operator/machine dialogue facilities.

The algorithm of the new master program is shown in Figure 5.21. The first action of the package is to give the operator the opportunity to modify the basic software configuration. In this newer version, virtually all the information on such things as input/output channel assignments and conversion factors, details of engine constants (such as torque to BMEP conversion factor), overspeed limits, which data parameters are to be logged and plotted and what, if any, correction factor is to be used, is stored in the form of data tables rather than being part of actual programs. As a result of this, it is possible to adapt the package for different hardware set-ups at execution time. If the operator requests it, the computer enters a subroutine which allows him to generate a new set of data tables.

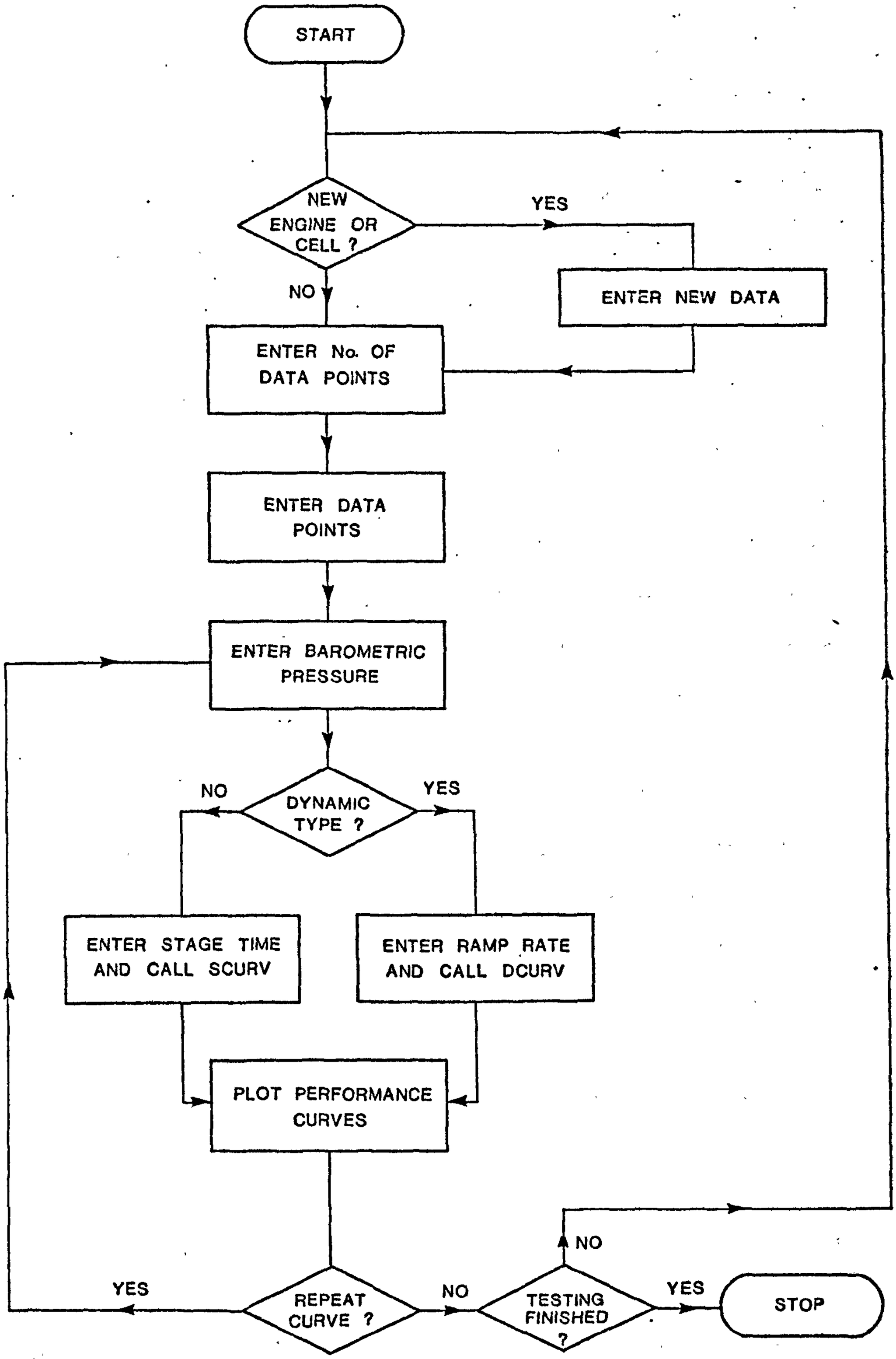


FIGURE 5.21 - EVALUATION PACKAGE MASTER ALGORITHM

The next stage is to set up the values of engine speed at which data scans are to be made. The computer requests the number of points (up to 40 are allowed). followed by a list of the speed values (see Figure 5.22). The barometric pressure reading must also be entered by the operator.

At this point the decision is made as to whether the full-load power curve to be run is going to be dynamic or steady-state in nature. If a dynamic type is requested the computer obtains the required ramp rate in rev/min/min and passes control to the dynamic sub-package (titled DCURV), or, for a steady-state request, inputs the duration of settling time for each stage in seconds, and goes to the steady-state sub-package SCURV. Both sub-packages have had their graph plotting functions deleted. Instead, on returning to the master program after completion of the actual test, an enhanced special plotting routine incorporating X-Y plotter calibration features and multiple copy plotting abilities, is called to perform the graphical presentation of the test data regardless of which type of test it originated from. Finally the operator can decide to repeat the same curves (but change from dynamic to steady-state or vice versa should he wish to), to start a new set of curves or to terminate testing completely.

The sub-packages have many 'invisible' changes which are not apparent to the user. Many of these were connected with the change over to using data tables to configure the system, and also to ensure that duplication of programming was avoided by allowing both packages to make use of the same basic common subroutines to perform such tasks

DYNAMIC POWER CURVE EVALUATION PACKAGE

```
NEW ENGINE OR CELL CONFIGURATION
?NO

ENTER CURVE DATA POINTS

?  09
?0800.
?1000.
?1200.
?1400.
?1600.
?1800.
?2000.
?2200.
?2400.

 BAROMETRIC PRESSURE ?
?76.0

 DYNAMIC TYPE ?
?YES

 ENTER RAMP RATE, REV/MIN/MIN
?0500.
```

DYNAMIC POWER CURVE

```
 SINGLE RAMP ?
?YES

 ENGINE READY ?
 PAUSE 00

?

 ENTER PLOT SPEED INTERVAL
?0.

 READY TO RAMP
 PAUSE 01
```

FIGURE 5.22 - OPERATOR / COMPUTER DIALOGUE

as set-point output, single point data logging and data reduction. Another major change which is not usually visible to the user, is the incorporation of a software malfunction monitor. This is done using the Operations Monitor Alarm facility in the computer hardware. Once activated, this system requires the execution of a special instruction, Pulse Monitor Alarm (PMA) at least once every 100 milliseconds. Failure to do so will result in processing being halted within the computer and an external signal line being activated. This is connected to the test-stand hardware and causes an automatic emergency shutdown of the test-bed. In the new package the real-time clock interrupt system is active all of the time. Normally, clock interrupts are serviced by a very simple routine which executes the PMA instruction and returns to the interrupted program. When the dynamic power curve executive becomes active it 'steals' the interrupt vector from this routine and points to itself. Its own service routine now includes the PMA command and on completion of the dynamic test, the previous interrupt vector is restored. This also happens if a dynamic test is aborted by an emergency shutdown occurring.

Except for the changes already mentioned in this or the previous chapter, there were no other alterations to the steady-state part of the package. The dynamic power curve section however had several. As has been mentioned, it was planned to investigate the results obtained by using double ramps. Thus the operator is given the opportunity to choose either a single or a double ramp test. In the case of the single ramp, an acceleration ramp is run from the lowest data logging speed up to the highest. For the double ramp,

the same acceleration ramp is performed but then the engine speed is allowed to continue to increase for a short period of time. At this point the direction of the ramp is changed and the engine begins to decelerate down through the data logging points until the starting speed is reached. The overshoot at the top of the ramp is intended to allow the engine to settle on the deceleration ramp after the disturbance of the changeover point.

Another change has also been made in the method of setting the interval between the high frequency logging points (minor logs). This is now done by an operator/machine dialogue which asks the operator to enter the desired interval in terms of rev/min. Should only the major log data be used in the graphical presentation, (to make the data obtained identical in form to that of the steady state type), specifying a zero or negative interval will disable the high frequency logging function. It is also possible to have the minor log include a full or partial scan of the other test bed parameters besides speed and torque (except for smoke readings).

One problem which did not arise in the earlier experiments was that of taking smoke readings as the engine used was a petrol engine. For the work at Dunton however it was proposed that diesel engines should be the subject and that their smoke emissions should be investigated. The difficulty lies in the fact that the type of smoke meters installed at Dunton are of the AVL type which use a cyclic measuring technique. The actual sampling period when gases are drawn from the exhaust is 1.5 seconds and it was felt that if this sample window could be accurately centred on the required logging speed, this

type of instrumentation could be used.

To accomplish this, several modifications were made to the major logging routine and its scheduling routine (LOG2 and TSPACE). The way this system works now is that TSPACE, instead of aiming to re-activate itself at the next logging point minus 50 rev/min, calculates the speed which should represent the engine speed 10.75 seconds before the logging speed is reached. It then uses this value as its aiming point. The normal test and delay sequence follows between activation and this new speed being reached. After this the sequence of events is as shown in Figure 5.23. The signal to start the smoke reading is sent by LOG2 via one of the computer digital outputs. The program then shuts itself down for 10.75 seconds. On reactivation it immediately performs a data scan of all the data inputs except the smoke reading. At this point the smoke meter is half-way through drawing its sample from the exhaust pipe and the engine speed should, if the ramp is being followed properly, be close to the desired logging speed. The routing must then shut itself off for a further 5 seconds whilst the smoke meter completes its internal operations and then presents the reading as an analog voltage to the computer interface. On its second reactivation the LOG2 routine takes this item of data and inserts it in the previously recorded information from the rest of the data scan before finally shutting itself down to await the next logging point operation.

This technique enabled useful data to be obtained on the behaviour of smoke emissions in dynamic as opposed to steady state conditions using the available instrumentation. However the drawbacks

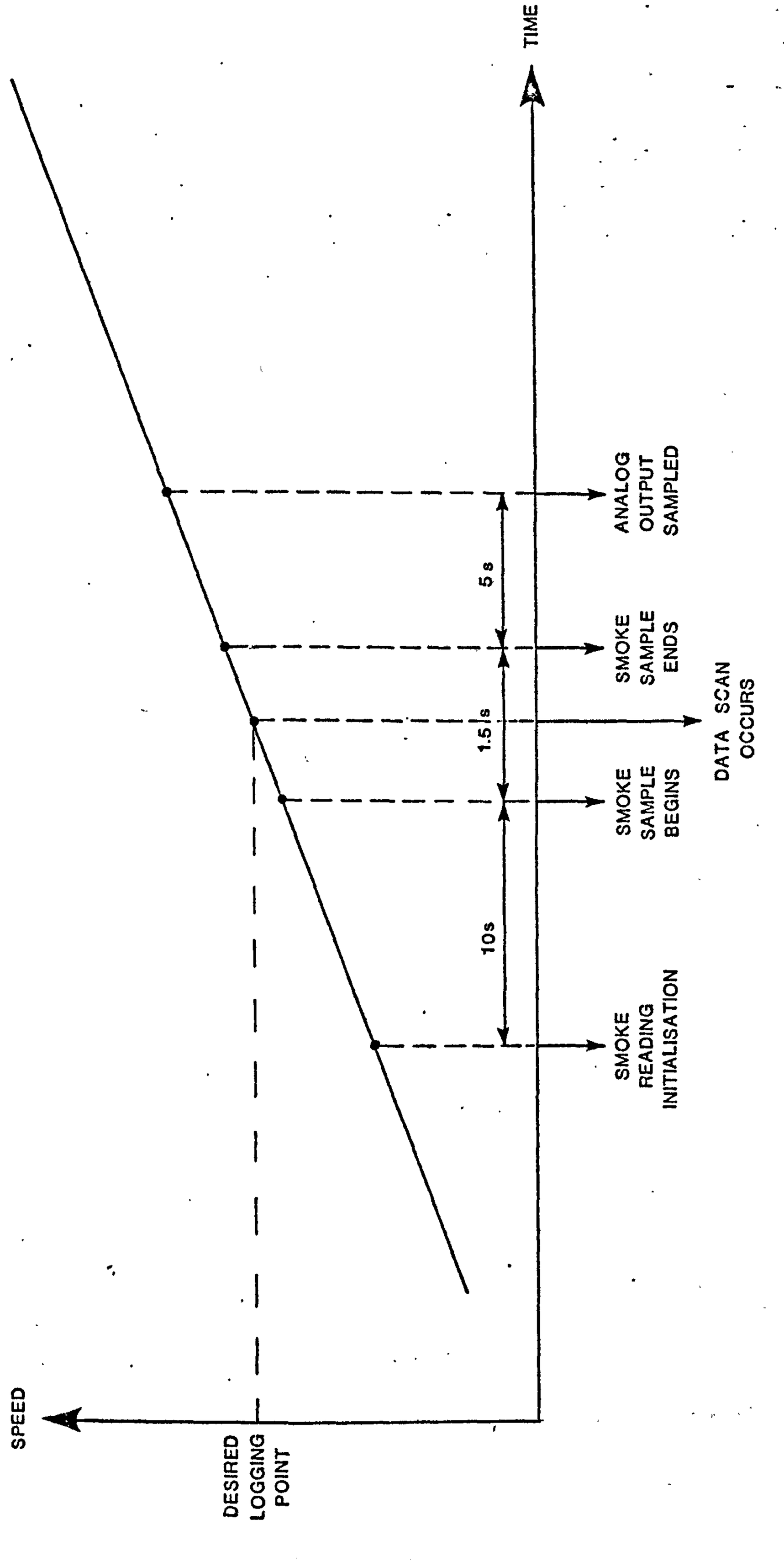


FIGURE 5.23 - SMOKE READING SYNCHRONIZATION

of such a method were that the accuracy with which the actual speeds at which logging occurred in relation to the desired values, suffered degradation, and that a limit was placed on the combination of maximum ramp rate and minimum interval between logging points. Obviously these values must be such as to allow at least 20 seconds between successive logging points occurring to allow the meter to complete its measuring cycle. These facts, added to the need for more complexity in the software, means that in any facility in future, in which it is planned to perform dynamic testing of diesel engines, should be fitted with instantaneous reading smoke meters rather than those of the cyclic measurement type.

5.4.2 - Evaluation Results

Initially the test bed was fitted with a six cylinder indirect injection engine of 3.5 litres capacity. The planned approach to the investigation was to run three different types of tests as follows:-

1) Manually controlled steady state

2) Computer controlled steady state

3) Computer controlled dynamic

Comparing our first category against the second would verify the correct function of the computer controlled set-up in general. Comparison between the second and third types would give an indication of the result of using dynamic methods.

The testing of the first engine was terminated due to breakdown after only a few tests had been completed. At this stage the final version of the software package had not been fully de-bugged

and so an intermediate version which lacked smoke data logging facilities was used. Nevertheless some interesting and encouraging results were obtained. Figure 5.24 shows a typical comparison of a manual steady state curve against a computer controlled steady state curve. Proceeding from there Figure 5.25 shows the results obtained from a dual ramp dynamic power curve run at a ramp rate of 500 rev/min/min and a steady state test under computer control with a stage time of 3 minutes. These results show that taking either dynamic ramp yields low errors and that use of the dual ramp averaging method would improve the correlation still further. These results showed that the dynamic technique can be applied to a variety of engines and still prove to be a valid technique.

The second engine to be used on the Dunton test bed was a six cylinder, turbocharged, direct injection diesel with a capacity of six litres. It was felt that this would prove to be the most difficult test of the dynamic method yet with its larger inertia and the effects of the turbocharger hysteresis. When compared with the original use of a 1.3 litre petrol engine, it would show the behaviour of the testing technique over a wide range of engine types and sizes.

An extensive program of tests of all the three different types was performed. Initally when analysing the data obtained, problems were experienced with poor repeatability. This applied to all three testing methods and could not be isolated to either the engine or the instrumentation. It became apparent that a major cause was due to variations in ambient temperature. Originally the standard practice of not applying any correction factor has been used. Normally

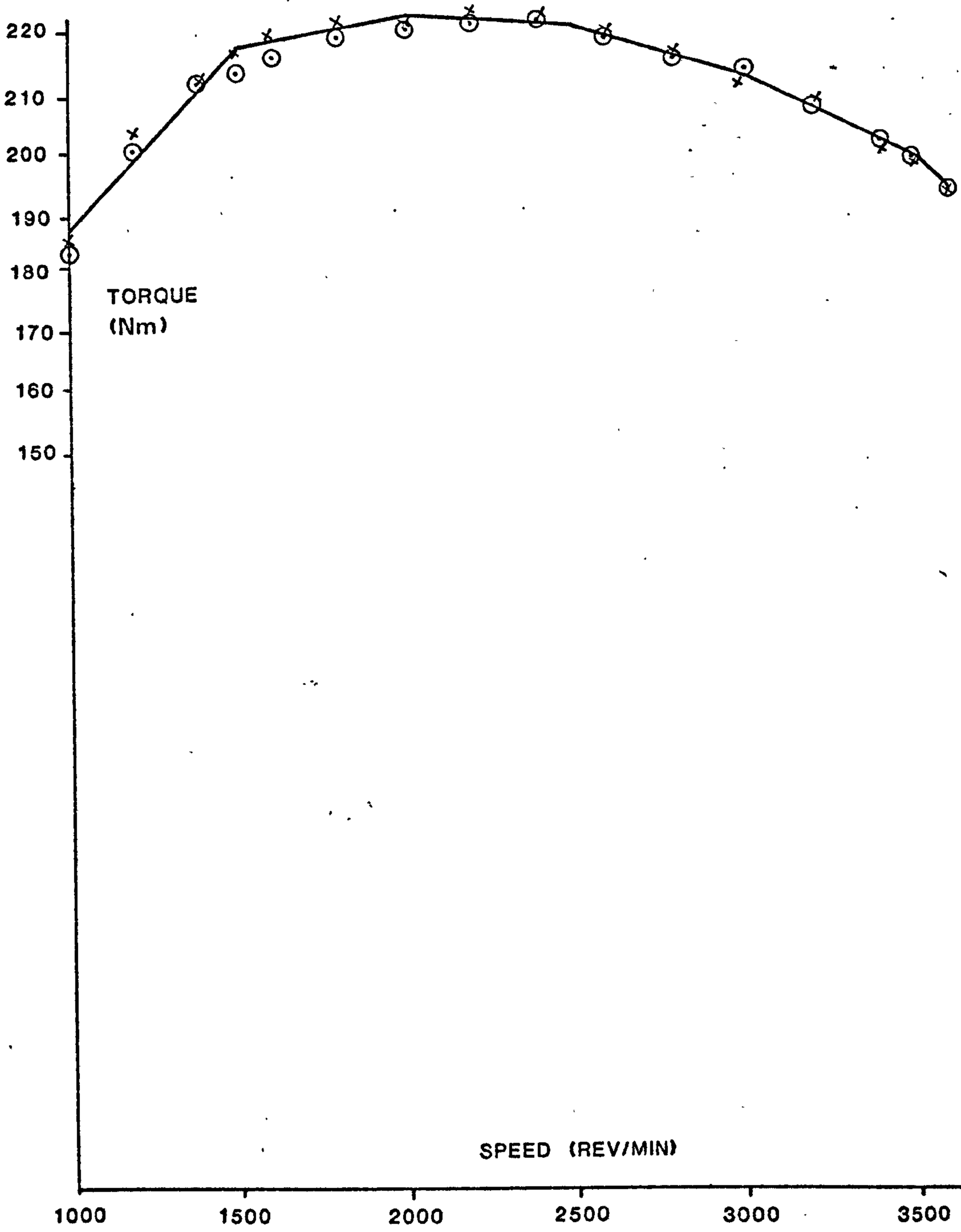


FIGURE 5.24 - YORK 6 FULL LOAD TORQUE

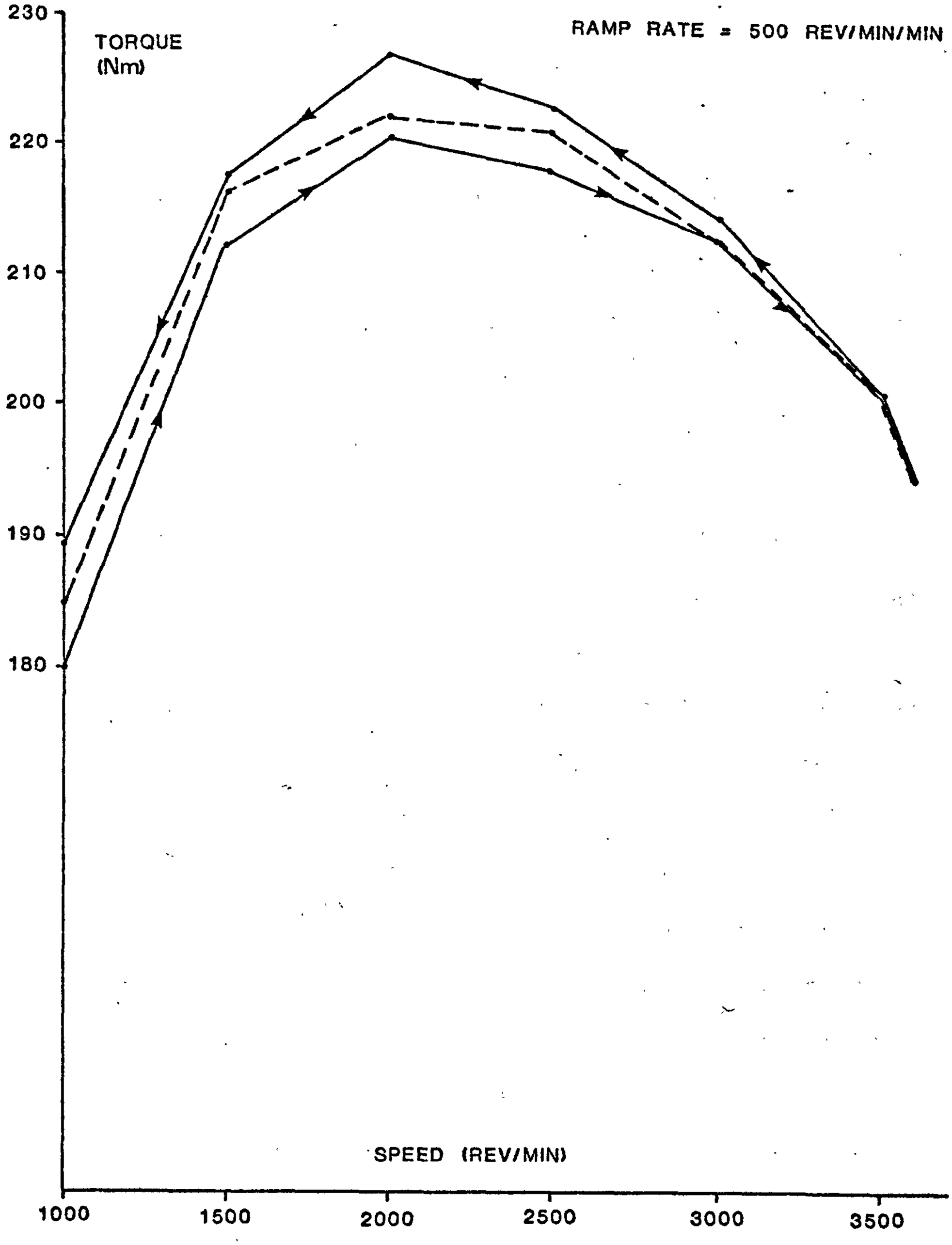


FIGURE 5.25 - YORK 6 FULL LOAD TORQUE CURVES COMPUTER CONTROLLED DUAL RAMP DYNAMIC AND STEADY STATE

however, the practice is to run all tests on turbocharged diesel engines at a constant air temperature (27° C). However the very high air temperatures experienced during the period (in summer) when the tests were performed, meant that the normal cell ventilation controls were unable to maintain the required temperature stability. It has already been shown that direct injection turbocharged engines vary greatly in torque output, with changes in air inlet temperature (17) with figures of up to 3% loss in power due to an increase of 10° C. Smoke density readings are also shown to be affected by changes in air temperature..

As a result, a special series of power curves were run to investigate the correlation between air temperature and variations in smoke and torque readings for the engine under test and to develop correction factors which could be applied to the collected data. The results obtained showed that torque losses of 7.5% or greater could occur with a 10° C rise in temperature. The actual variation appeared to be fairly constant throughout the speed range and the correct factor taken from the results gave a loss of 50 kPa in brake mean effective pressure for a 7° C increase in inlet air temperature. In the case of the smoke readings the findings were not quite so simple. It was decided that four different speeds were to be used to evaluate the data. These were the speeds of maximum power, 2400 rev/min, (40 rev/s), maximum torque, 1800 rev/min (30 rev/s), and two lower speeds, 1400 and 1000 rev/min (23.3 and 16.7 rev/s). Unfortunately the air temperature/smoke density relationship was not constant for all four speeds and the following relationship was instead used as the basis for data correction:-

Speed (rev/min)	Variation in Smoke Density (Bosch Smoke Units/10° C)
1000	+ 0.6
1400	+ 0.8
1800	+ 0.4
2400	+ 0.4

The above factors were then applied to correct the torque, BMEP and smoke readings.

The results obtained for ten tests at each acceleration rate were analysed. The rates used were 0 (Steady-State), 200 rev/min/min (0.37 rad/s^2) and 500 rev/min/min (0.87 rad/s^2). For higher ramp rates the results showed unacceptable lack of repeatability, particularly at the low end of the speed range.

The results for brake mean effective pressure are shown in Figure 5.26. At each of the four speeds the variation in BMEP is plotted as a function of its percentage deviation from the mean steady state value. The vertical lines represent the scatter between the maximum and minimum results obtained during the 10 tests and the horizontal lines are drawn through the mean values for all tests. All results are for acceleration ramps only. The graphs show that the mean values vary by less than ± 1% for all speeds and acceleration rates and that scatter is not significantly worse for dynamic conditions than it is for steady state ones.

The results shown in Figure 5.27 for the smoke readings are not as encouraging. There is a large amount of scatter for all

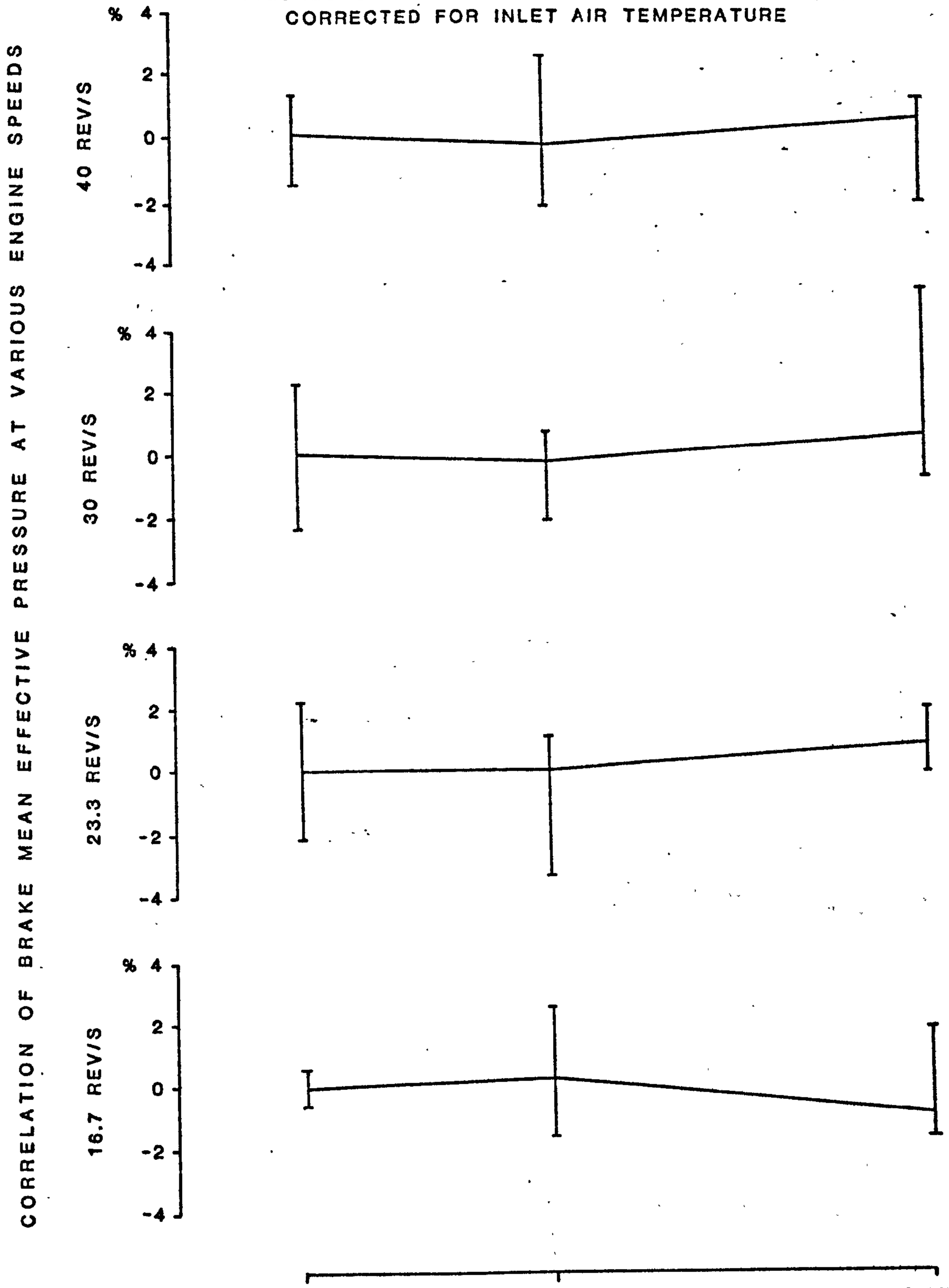


FIGURE 5.26 - ENGINE PARAMETERS OBTAINED AT VARIOUS RAMP RATES

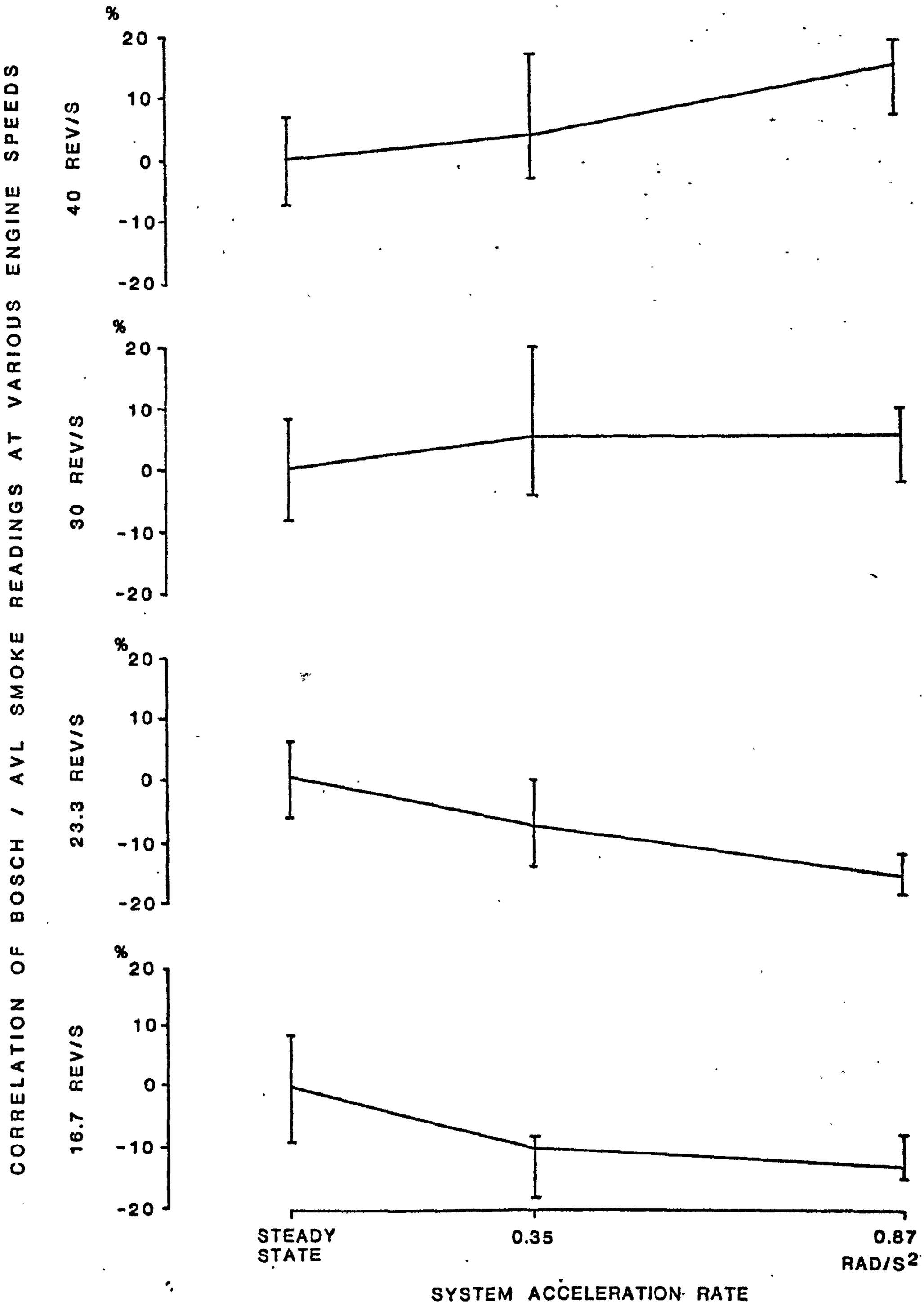


FIGURE 5.27 -ENGINE PARAMETERS OBTAINED AT VARIOUS RAMP RATES

the tests which makes accurate analysis difficult. Nevertheless the overall trend is still apparent. At low speeds dynamic effects tend to decrease the smoke density, whilst at high speeds the opposite is true.

Turning to Figure 5.28 which shows a plot for both acceleration and deceleration ramps this trend is clearly shown and a closer correlation with steady state values, taken over the whole speed range, could probably be achieved by using the deceleration ramp smoke readings.

Turning to the other data parameters, the boost pressure shows a typical hysteresis effect with close correlation between dynamic and steady state results. An even better fit could be obtained using the dual ramp averaging technique. Figure 5.29 shows this technique extended to some of the other data parameters with the resultant good correlation.

5.5 Summary

The chapter shows the way in which a computer controlled test rig can be programmed to perform dynamic type full load power curve tests. The technique was applied to a wide range of engine types and the results obtained evaluated. The early part of the experimental programme showed that good correlation with steady state power data could easily be obtained with small petrol engines and indirect injection diesel engines. In the case of a large turbocharged direct injection engine the problem became more complicated. With the application of an air temperature correction factor, good correlation could be shown in most parameters by using either the accelera-

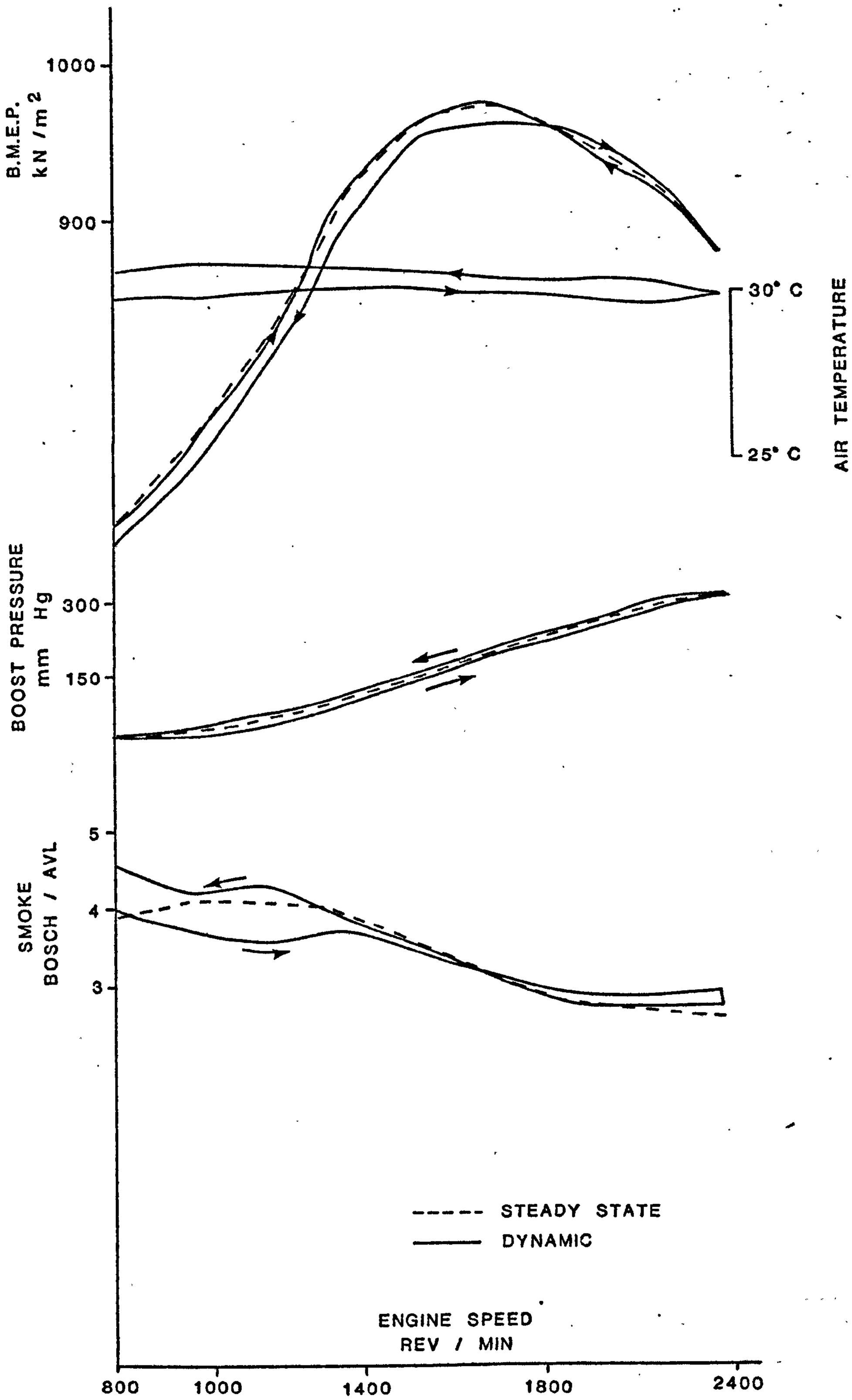
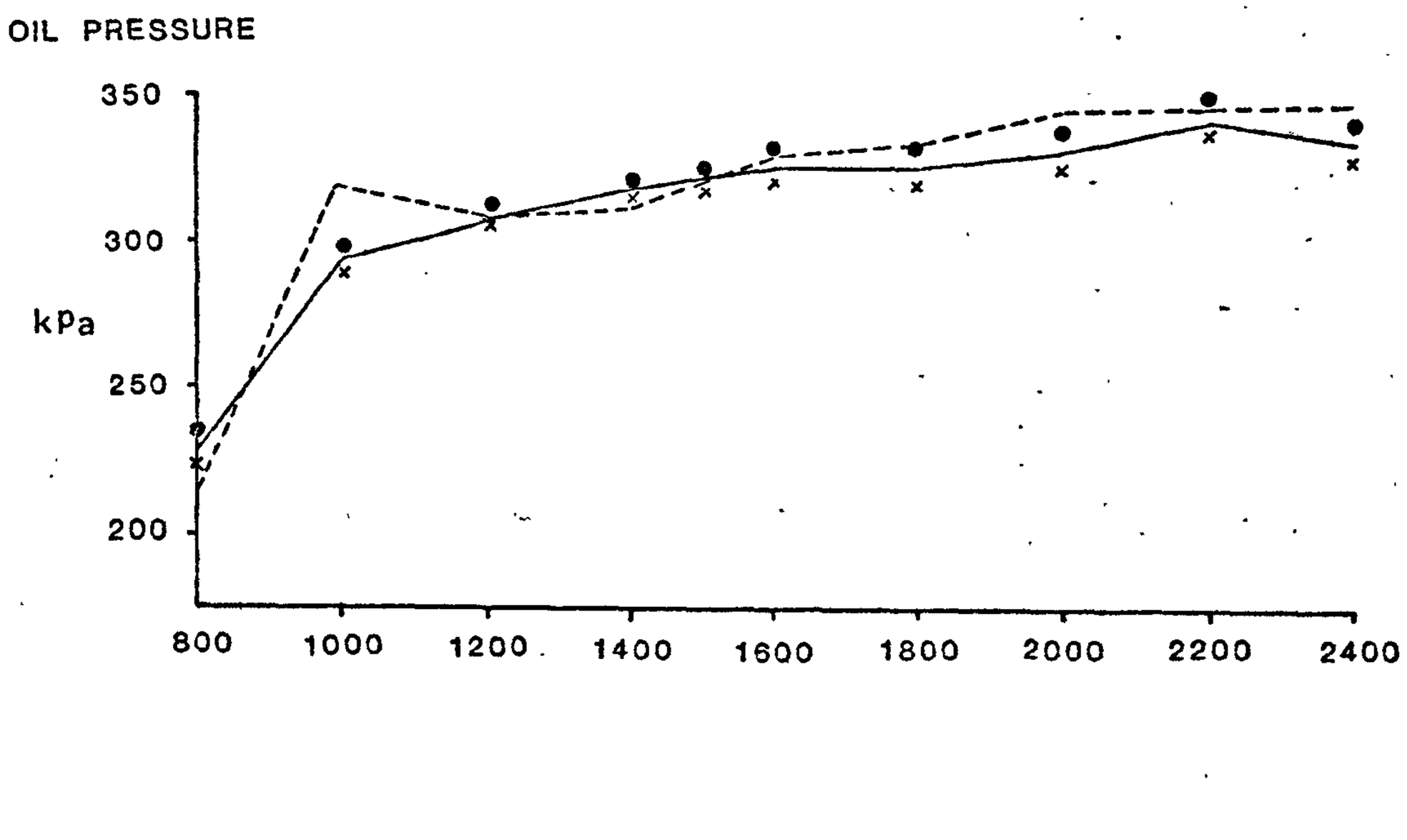
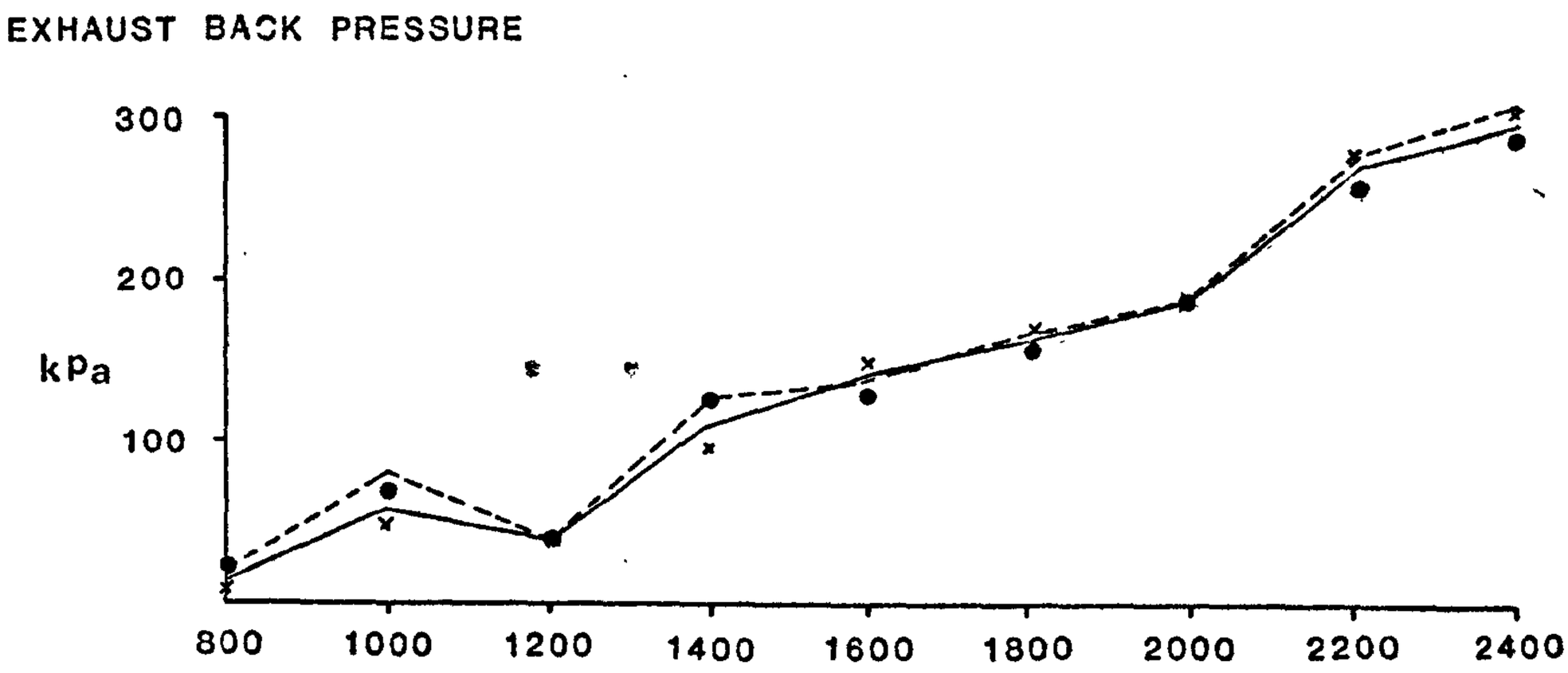
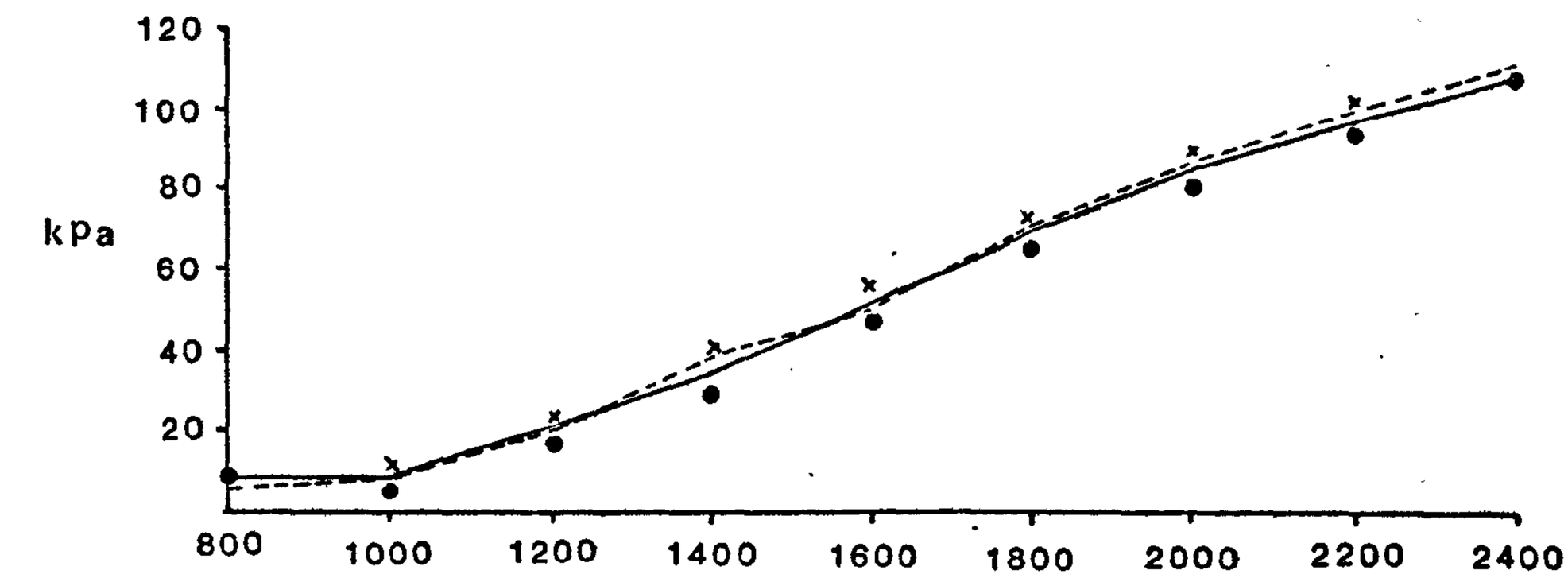


FIGURE 5.28 - COMPARISON OF A TYPICAL STEADY STATE POWER CURVE WITH A POWER CURVE OBTAINED USING A 0.87 RAD / S² (500 RPM / MIN) RAMP RATE







● = ACCELERATION RAMP
✕ = DECELERATION RAMP
—— = DUAL RAMP AVERAGE
--- = STEADY STATE

FIGURE 5.29 - PRESSURE DATA

tion or the average of a dual ramp test to compare with steady state values. The exception to this rule was the smoke density data which it was difficult to correlate.

It is possible to argue that the smoke readings obtained from the dynamic tests present a truer picture of engine behaviour during normal usage where dynamic rather than steady state conditions tend to be prevalent, particularly in connection with urban driving. Such a consideration would seem to be borne out by the use made of dynamic states to test engine smoke emission as described in the next chapter.

Unfortunately it was not possible to investigate fuel consumption behaviour under dynamic conditions due to the lack of a suitable instanteneous reading flow meter. With the proviso that such results can also be correlated with steady state data, it is suggested that the dynamic power curve can be considered an alternative to steady state techniques. In the short term it can result in test time savings giving improved economic justification for automated testing. For example in the final tests described a dual ramp power curve at a ramp rate of 500 rev/min/min took about 6½ minutes. An equivalent steady state test taking readings at 200 rev/min intervals and having a 3 minute stage time took almost 30 minutes to complete.

In the future it will be possible to extend dynamic testing to other techniques such as part-load curves and governor curves. The advantage of savings in testing time may well be augmented by the collection of data that is more realistic in terms of common engine usage and behaviour.

Chapter 6

CYCLE TESTING OF ENGINES

6.1 Introduction

The concept of a repeated test cycle in connection with the testing of internal combustion engines has achieved considerable importance in the light of the increasing amount of legislation connected with emission levels. The idea of such a test cycle is to demonstrate the emission performance of an engine to ensure that it can meet the levels currently regarded as acceptable.

There are two prime reasons for the use of a high level automation strategy to accomplish this testing. Firstly, in an attempt to simulate accurately the conditions which will exist in normal engine usage, many of the basic test cycles are dynamic in nature using a series of high acceleration ramps in engine speed. The capabilities of the operator of a conventional manually controlled or semi-automated test stand are too limited to achieve satisfactory running of the required test cycle.

The second point arises from the fact that it is vital that the method of testing is accurate and highly repeatable. If many different engines are going to be tested on differing test stands, considerable effort must be made to ensure that the results obtained are compatable with one another or the whole concept of a standard requirement is impossible. The natural human tendency to vary in behaviour and response from day to day has been shown to be one of the major causes of error during manually controlled tests (38). Thus in setting out to investigate the needs for the design of tomorrow's automated engine testing, it was felt that considerable importance should be attached to providing the capability to run dynamic test cycles.

Whilst there are a great many different emissions tests used or planned throughout the world, many are quite similar, if not from a detailed point of view, then certainly from the basic control aspect. The work in this chapter is centered on two tests, both from the United States of America and concerning heavy duty diesel engines. These tests were among the first to be defined and to a great extent set the pattern with the other tests appearing throughout the world often being highly derivative in concept. One of the test cycles is used for measuring smoke opacity whilst the other is concerned with the actual gaseous emissions. It is felt that the smoke cycle for heavy duty diesel engines in particular provides an especially stringent test of the controlling system's abilities.

The following section of the chapter provides more detail of the tests themselves. It is primarily concerned with the two tests mentioned above but others selected on an international basis are also included for comparison purposes. As with most legislative information, the complete scope of the regulations passed is immense and so the actual tests themselves are concentrated on such matters as the techniques of vehicle sample selection and statistical verification of results are passed over.

Subsequently the work done on developing test systems capable of performing the required cycle is described. It should be emphasised that the object of the work related to the control of the engine and dynamometer, not the actual resultant emissions. The work on the control aspect is vital to the future of the automation of engine testing, the problem of minimising harmful emissions is the concern

of the engine designers and hence falls outside the scope of this thesis.

6.2 Emission Regulations for Diesel Engines

6.2.1 - United States of America

The United States Environmental Protection Agency (U.S.E.P.A.) is responsible for legislation covering the emissions from all forms of motor vehicles. As stated our main concern here is with heavy duty diesel engines. The U.S.E.P.A. Smoke Cycle is shown in Figure 6.1. (2). The test cycle consists of three distinct modes.

1) Idle Mode - The engine is allowed to idle for 5 to 5.5 minutes. The dynamometer load must be minimised if it is not possible to switch the load off entirely.

2) Acceleration Mode - This mode consists of four different sections:-

i) The engine speed is increased by 200 ± 50 rev/min above the idle speed. This must occur within 3 seconds.

ii) The engine shall be accelerated at fully open-throttle against a dynamometer load so that the engine speed reaches 85 to 90% rated speed in 5 ± 1.5 seconds. The acceleration ramp must be linear to within ± 100 rev/min.

iii) On reaching the above speed the throttle must be rapidly closed and the engine speed shall be reduced to the maximum torque speed or 60% rated speed, whichever is higher. The tolerance on this point is ± 50 rev/min. The smoke emissions during

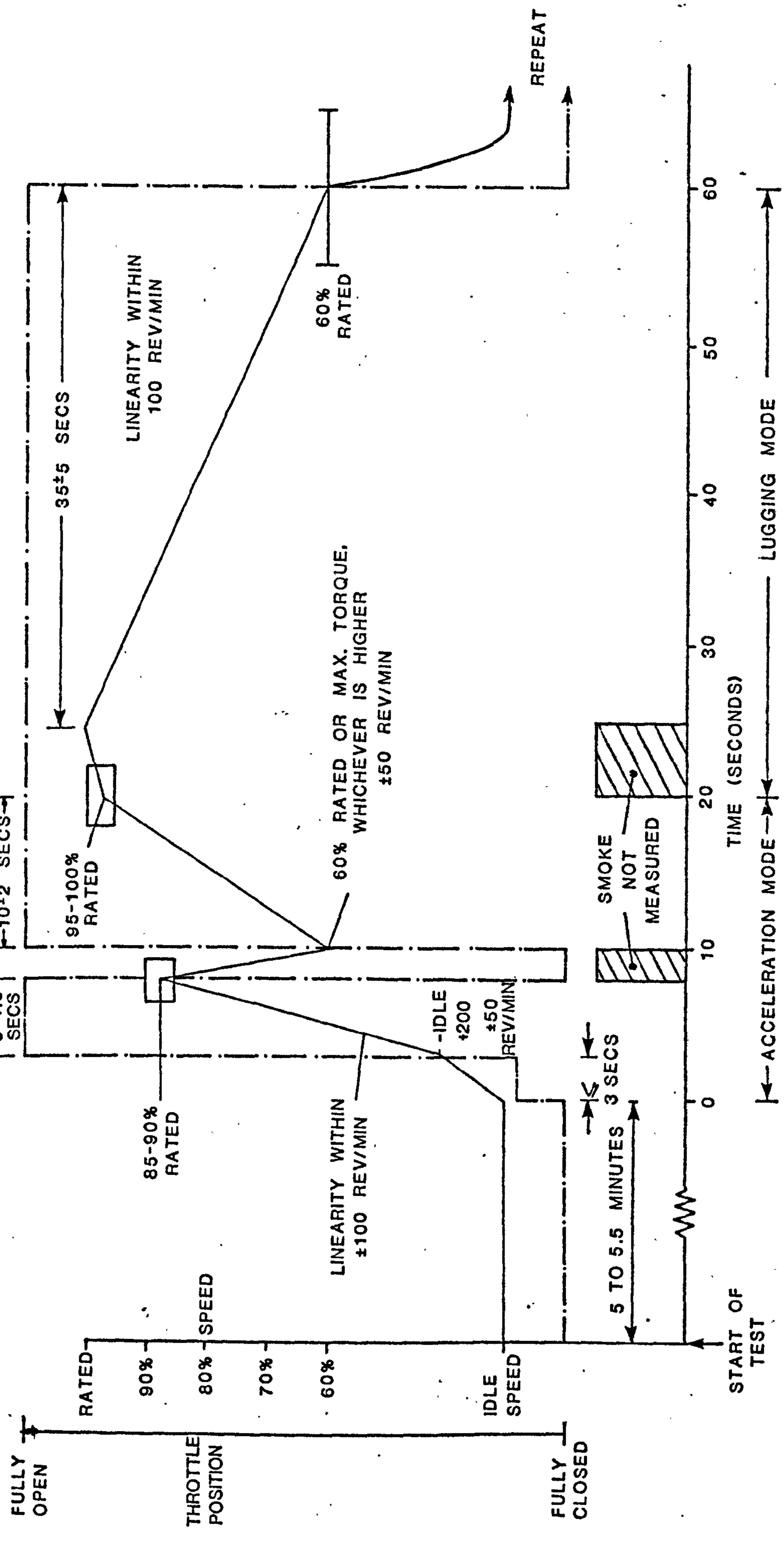


FIGURE 6.1 - E.P.A. SMOKE CYCLE

this stage are not used in determining the test results.

iv) The throttle is then moved to the full open position again so that the engine accelerates against a dynamometer load to reach 95 to 100% rated speed in 10 ± 2 seconds.

3) Lugging Mode - Following on from the acceleration mode the controls should be adjusted to allow the engine to develop maximum power at rated speed. Again smoke emissions need not be measured during this stage. The engine is then decelerated at full throttle by the application of load. The speed should then be reduced to the maximum torque speed or 60% rated speed, whichever is higher in a period of 35 ± 5 seconds. The deceleration should be smooth so that the slope is linear to within ± 100 rev/min.

On completion of the lugging mode the engine is returned to the idle mode. In the proper certification test, the cycle is then repeated twice more.

The limits on the smoke emissions of an engine are defined in terms of the exhaust opacity (3). Thus the opacity must not exceed:-

i) 20% during the acceleration mode

ii) 15% during the lugging mode

iii) 50% during the peaks of either mode.

As a counterpart to the above regulations the U.S.E.P.A. has also issued requirements for new heavy duty diesel engine gaseous emissions (2). These require a different test cycle which, as shown in Figure 6.2, has much more in common with the steady-state method of testing. The cycle includes 13 modes, consisting of 3 idle modes

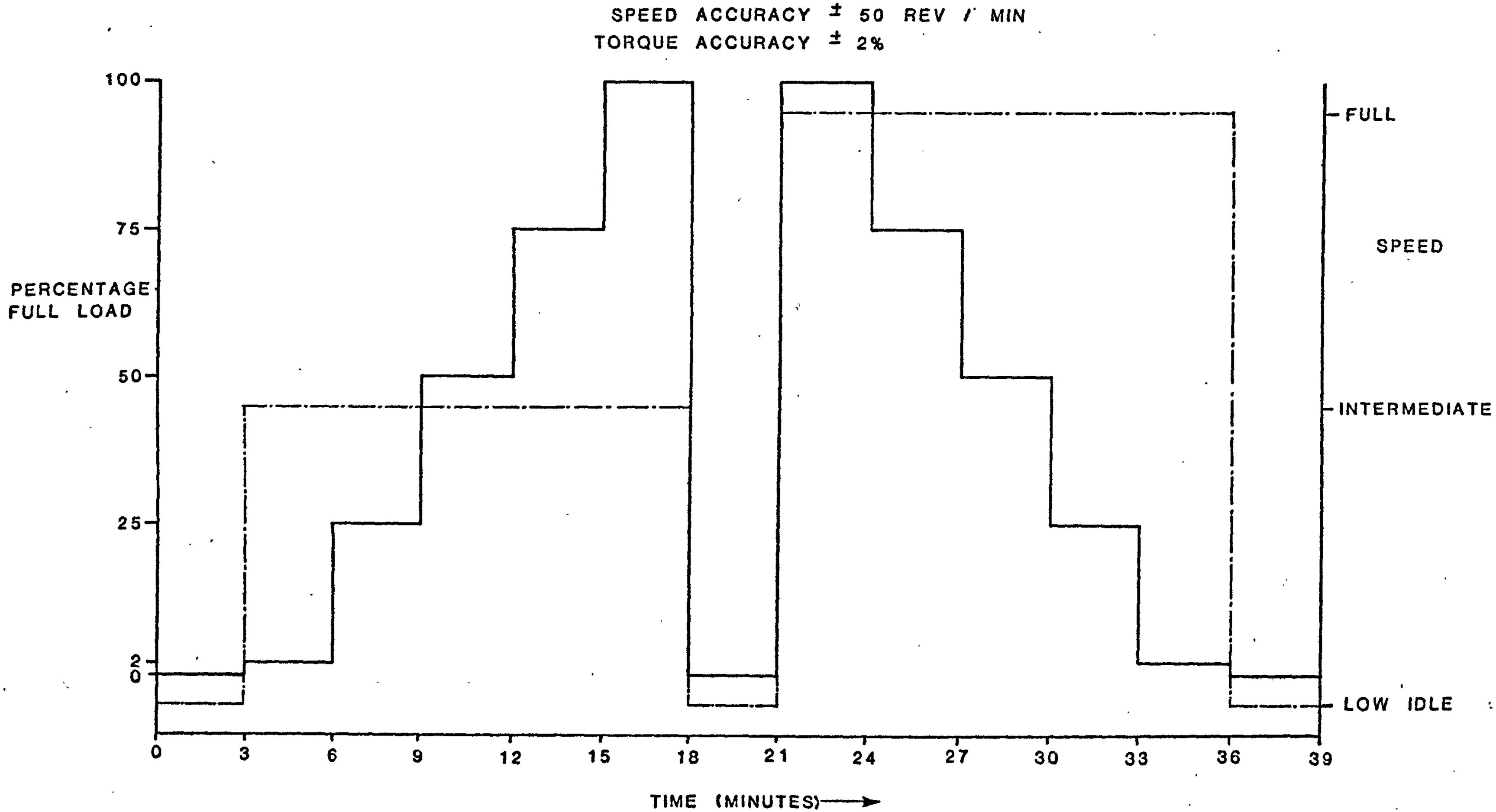


FIGURE 6.2 - THIRTEEN MODE CYCLE

and 5 modes with the torque at 2, 25, 50, 75 and 100% maximum load repeated for both intermediate and full rated speed. Each of the modes lasts for 3 minutes giving a total cycle time of 39 minutes. During each mode the speed must be held to within ± 50 rev/min and the torque to within ± 2%. The legal requirements state that the exhaust emissions during the test must not exceed:-

i) 16 grams per brake horsepower hour in the case of hydrocarbons and oxides of nitrogen.

ii) 40 grams per brake horsepower hour for carbon monoxide.

6.2.2 - European Economic Community

On behalf of the member states of the EEC the Council of the European Communitites has issued its directive on smoke emissions from diesel engines (1). This directive requires an engine or vehicle to be subjected to two separate tests. The first test consists of taking six measurements for full load (maximum torque) at engine speeds uniformly spaced out between the speed of maximum power and 45% of this speed, or 1000 rev/min whichever is higher. The smoke limits are defined in terms of absorption coefficient of the exhaust gases, against the nominal gas flow as shown in Figure 6.3.

The second test is known as the free-acceleration test. With the engine at idle the throttle is opened fully and the engine allowed to reach its maximum speed under free acceleration. At this point the throttle is closed again and the engine decelerates back to its idle speed. This sequence must be repeated not less than six times in all (see Figure 6.4). During the course of the

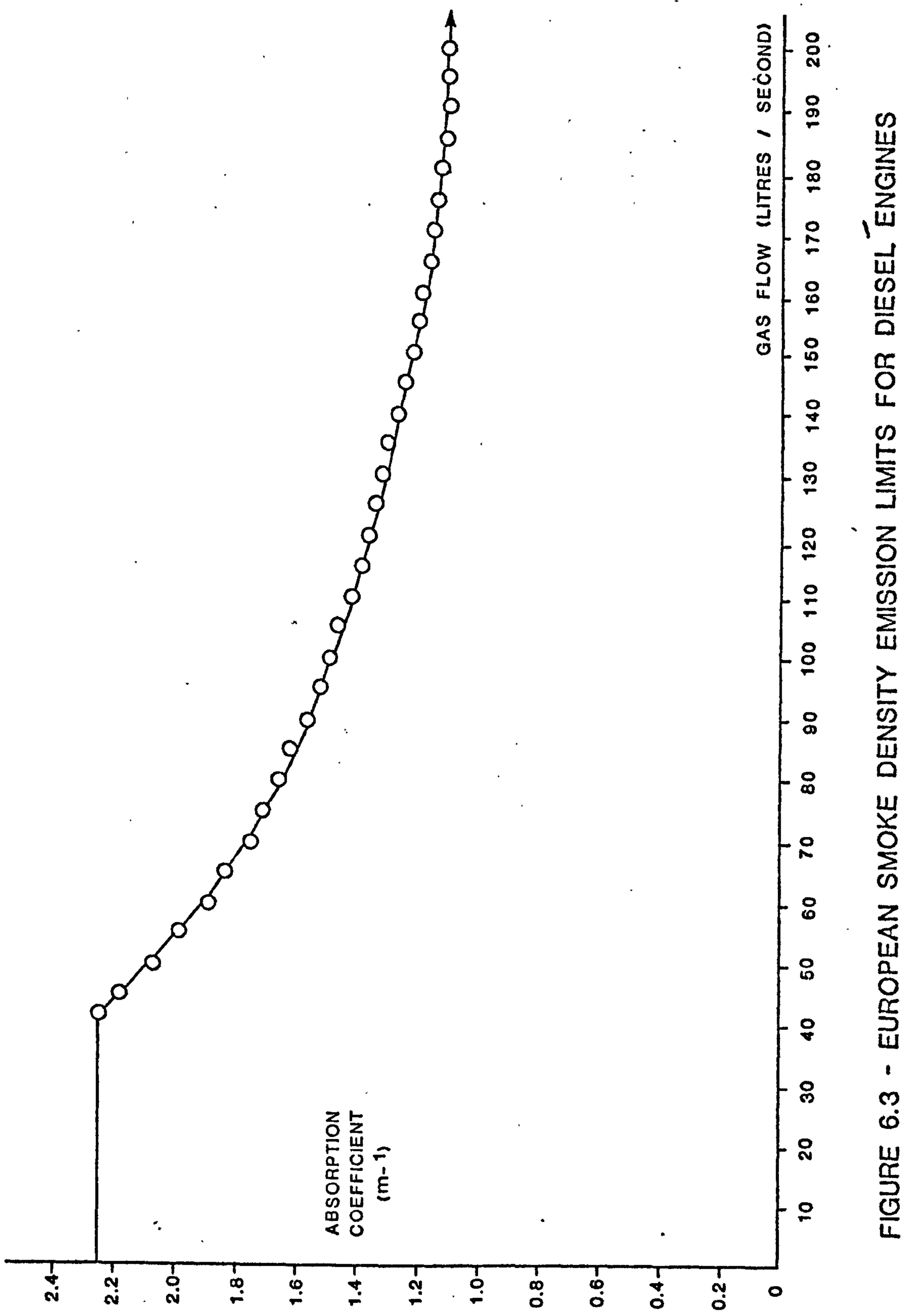


FIGURE 6.3 - EUROPEAN SMOKE DENSITY EMISSION LIMITS FOR DIESEL ENGINES

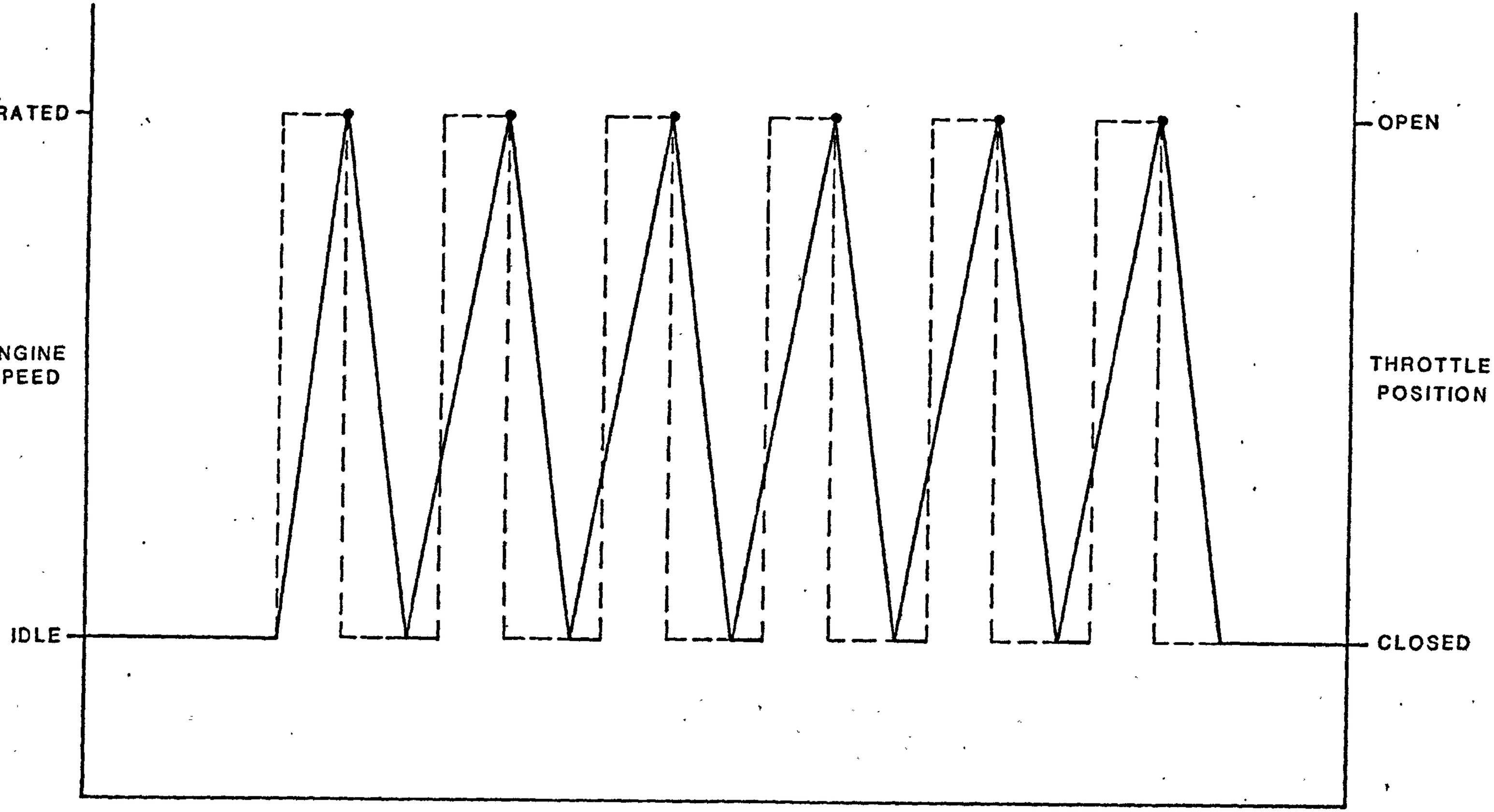


FIGURE 6.4 - E.C.E. FREE ACCELERATION TEST

test, the absorption coefficient should not exceed 0.5m^{-1} greater than the equivalent limit for the constant speed test.

6.2.3. - Sweden

Regulations put forward by the National Swedish Road Safety Board propose limits for the smoke emission from diesel engines (4). In the case of heavy duty diesel engines, the smoke density limits during a constant speed engine/dynamometer test are 2.5 Bosch units or 30 Hartridge units. The measurements should be made at full load with a speed greater than 50% rated speed. In addition the speed must not be high enough to incur any risk of the governor coming into operation. Thus a speed of between 50 to 70% of rated speed is recommended.

6.3 U.S.E.P.A. Smoke Cycle

The first series of tests were concerned with the EPA Smoke Cycle described in the previous section. The work was performed at Queen Mary College using one of the test rigs described in Chapter 3. The engine under test was a Ford Dorset 4 diesel engine. The strategy used at this stage involved relying to a great degree on the accurate performance of the test rig's local analog controllers, with the computer solely responsible for outputting the required set points.

The Federal Smoke Cycle for the Dorset Engine is shown in Figure 6.5. As the software package was intended to provide a

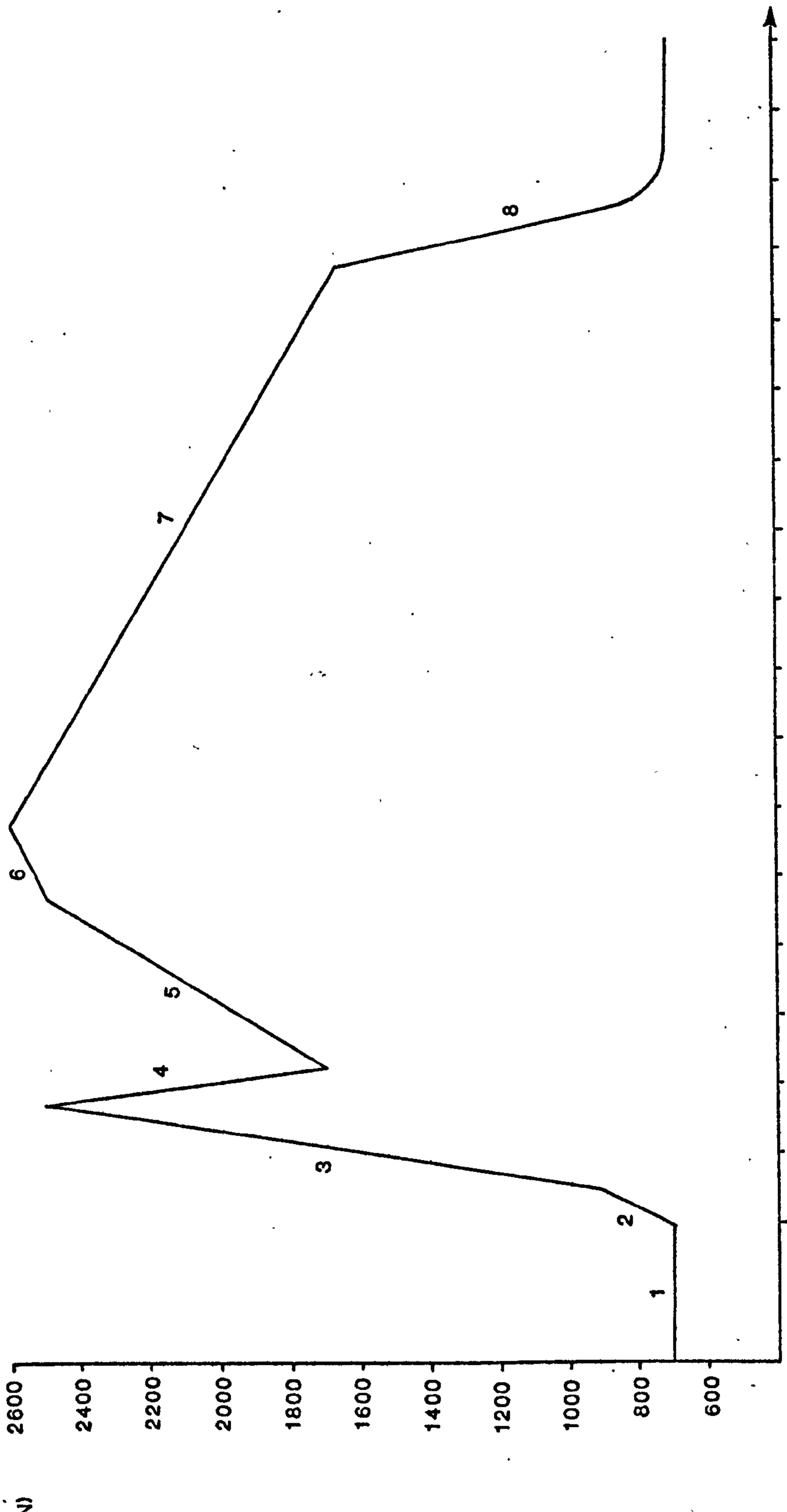


FIGURE 6.5 - DORSET ENGINE SMOKE CYCLE SPEED

more generalised test system, the various parameters needed to define the test cycle were to be read in at execution time.

6.3.1. - Smoke Cycle Package

The software package consisted of several special programs described below which were written in Fortran. In addition the standard Minicontroller and Fortran Libraries were used. The algorithm of the mainline routine is shown in Figure 6.6. After initialising the input/output system for the test rig concerned, the program then sets up the rig's local circuits for Mode III control, i.e. direct control of the throttle position and speed control by dynamometer loading, and prints out a heading message on the teletype (see Figure 6.7). The next stage is to obtain the data for the cycle test to be run which is normally accomplished by feeding a short paper tape via the high speed reader. This is done by a subroutine called STEST as shown in Figure 6.8 which then prints out the data to allow operator verification. This data, which is all that is necessary to define the test, takes the following format:-

a) Item 1 - the number of steps in the cycle.

b) Four parameters for each of the steps. These are:-

i) The end of step desired engine speed.

ii) The allowable tolerance on i)

iii) The throttle position

iv) The maximum duration of the step

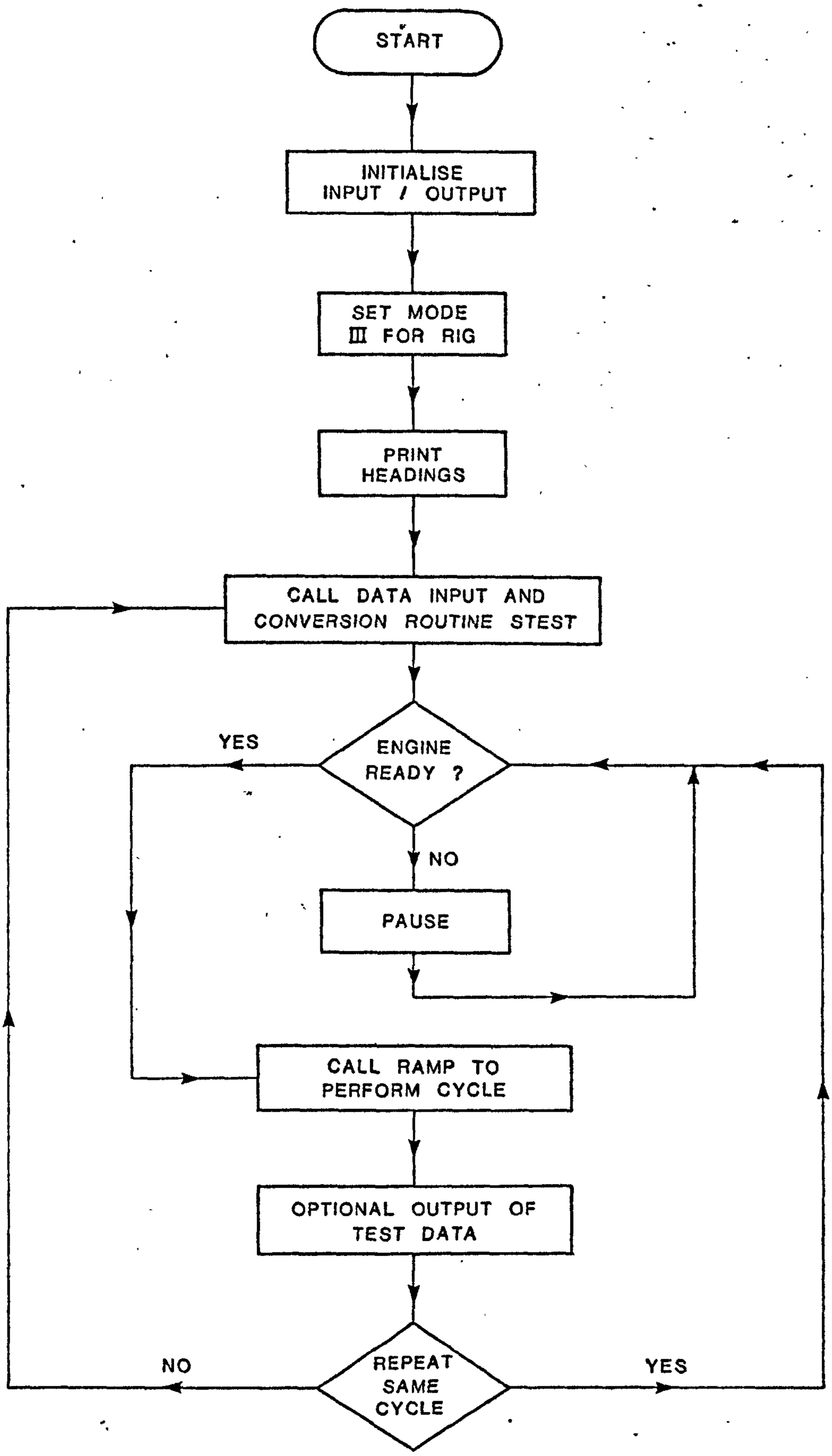


FIGURE 6.6 - MAIN CYCLE TEST ALGORITHM

```
Q.M.C.
DYNAMIC TESTING PROJECT

FEED DATA TAPE
PAUSE 00
?

VERIFICATION OF INPUT DATA
NO. OF STEPS = 8

 SPEED          TOLERANCE        THROTTLE          TIME
  RPM             RPM            % FULL            SEC
  700              0              0.00             0.0
  900             10              0.08             2.0
 2500             50              1.00             5.0
 1700             50              0.00             2.0
 2500             50              1.00            10.0
 2600             10              1.00             4.0
 1650             10              1.00            32.0
  700             10              0.00            10.0

 O.K. ?
?NO
 PAR. NO., STEP NO.
?1,8

 STEP 8
 SPEED = 700
?2800
 PAR. NO.,STEP NO.
?0,0

VERIFICATION OF INPUT DATA
NO. OF STEPS = 8

 SPEED          TOLERANCE        THROTTLE          TIME
  RPM             RPM            % FULL            SEC
  700              0              0.00             0.0
  900             10              0.08             2.0
 2500             50              1.00             5.0
 1700             50              0.00             2.0
 2500             50              1.00            10.0
 2600             10              1.00             4.0
 1650             10              1.00            32.0
 2800             10              0.00            10.0

 O.K. ?
?YES
 ENGINE READY ?
?YES
 RESULTS ?
?YES
 END OF TEST
 REPEAT WITH SAME DATA  ?
?NO
```

FIGURE 6.7 - TELETYPE OUTPUT

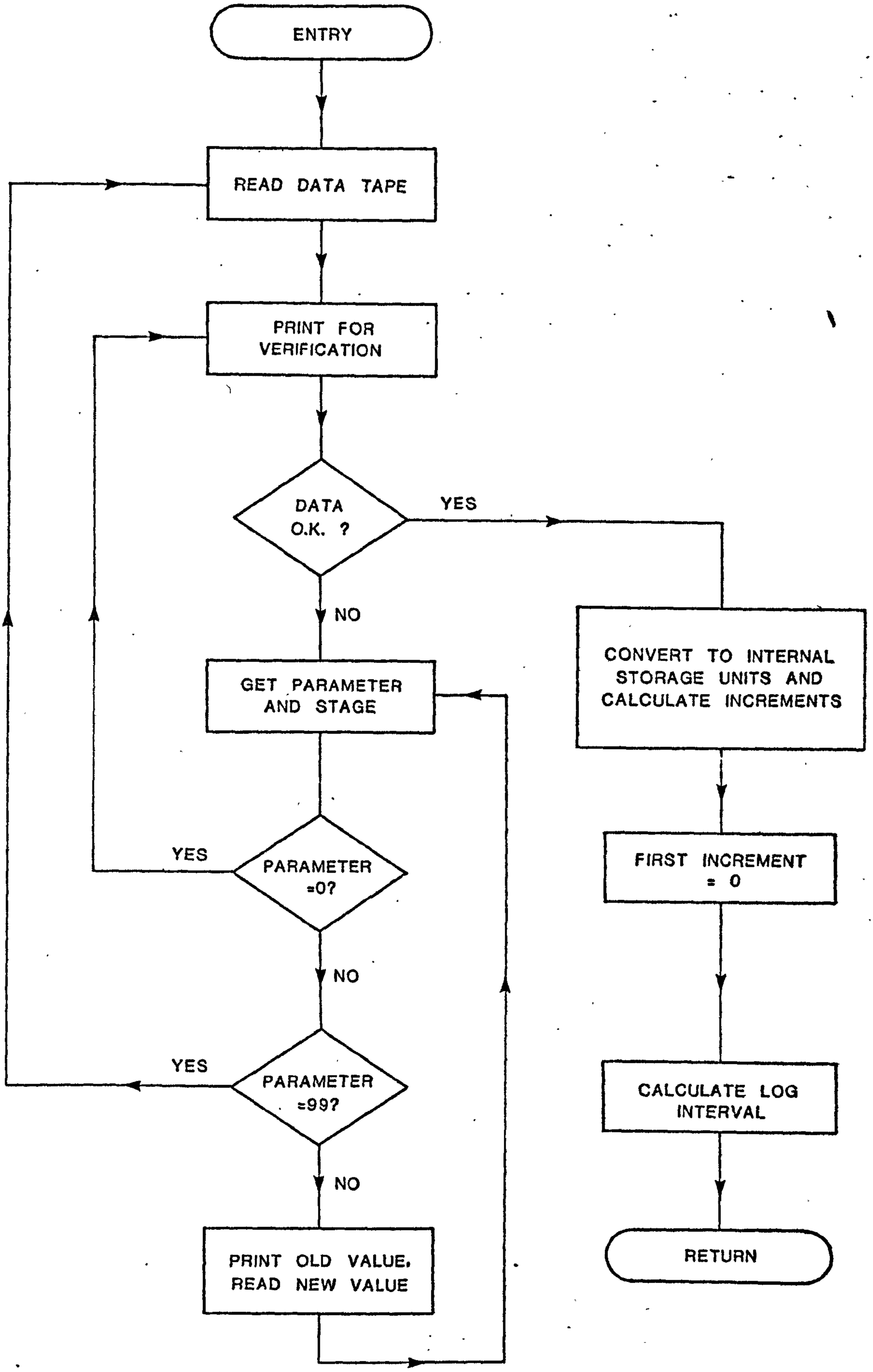


FIGURE 6.8 - STEST ALGORITHM

Each step is defined as a ramp in speed from the value at the end of the previous step to the desired value at the end of the current step. The throttle position is held constant throughout a step, i.e. any change requested by the data takes place as a step change at the start of the step. This value is maintained until a different one is required.

If we compare the table of values in Figure 6.7 to Figure 6.5, we can see how this data list is used to define the Federal Smoke Cycle for the Dorset engine. The number of steps is eight as numbered in Figure 6.5. For each step we have the speed defined in revolutions per minute with a plus and minus tolerance figure in the same units. The throttle opening is defined as a fraction of fully open (i.e. 0.08 means the throttle should be 8% open) and the step time is given in minutes to an accurracy of a tenth of a second.

When these values have been printed the operator is offered the opportunity of modifying any number of them. Should he indicate his wish to do so, the computer will ask for the parameter number and stage number which the operator must then enter. Thus if he types 1, 8 he is indicating that he wishes to alter the first parameter (speed) of the eighth stage. The computer will respond by typing the current value of the parameter and then obtaining the new value from the operator. This procedure will continue until the operator types a parameter number of zero or 99. The value of 99 will cause the computer to return to the stage of reading in a new data tape. The value of zero on the other hand returns to printing out the data parameters as they now stand for confirmation of

their verification.

This process of data parameter modification can be clearly seen in Figure 6.7. It is interesting to note that the change shown, the altering of the final speed setting from 700 rev/min to 2800 rev/min was in fact always used in the tests. Although it is true that the last leg of the cycle is defined as bringing the engine speed back to idle (700 rev/min for the Dorset engine), actually defining a speed control ramp to do this has unwanted consequences. At the beginning of the last step the throttle is suddenly closed having been fully opened. This alone is sufficient to cause the engine speed to drop rapidly back to idle. However if in addition the dynamometer loading is also increased to reduce the speed then the speed may drop very rapidly and because of the comparatively slow response of the dynamometer speed control loop , the engine may stall. To avoid this we use an 'artificial' speed ramp for the last step and rely on the throttle closure to perform the required speed change. This is perhaps a very simple introduction to the concept of a more adaptable use of the computer abilities which will play a much more important role later on in this chapter.

Returning to our consideration of the STEST algorithm, we can see that when the operator finally accepts the data parameters, the computer moves on to a series of conversion and calculation operations. The software package is operated on the basis that all parameters used during the test are stored directly in terms of the various voltage levels at the process interface rather than in engineering units so avoiding the need for any time consuming conversion mathematics

whilst the actual test is running. In addition the computer also calculates in advance the voltage step needed as a increment to the speed set-point at 10 millisecond intervals to generate the required demand ramp. Thus STEST creates a data array which contains six parameters for each step of the test. These are:-

i) The end of step speed as a voltage

ii) The end of step speed minus the maximum tolerance

iii) The end of step speed plus the maximum tolerance

iv) The throttle position as a voltage

v) The step duration in milliseconds

vi) The 10 millisecond speed increment as a voltage

The increment for ramping during the first step is defined as zero. This is because the first step is a dummy one to allow the rest of the test to proceed from the engine idle situation.

The test cycle program also includes the logging of performance data. 150 sets of data can be stored in an array and STEST also calculates the time interval required to distribute these values evenly over the test duration.

On returning to the mainline program the operator is asked if the engine is ready and when this is confirmed the actual test takes place as a result of a call to the subroutine RAMP whose algorithm is shown in Figure 6.9.

The first action of RAMP is to set up the various input/output channel assignments it will use, zero its internal software time

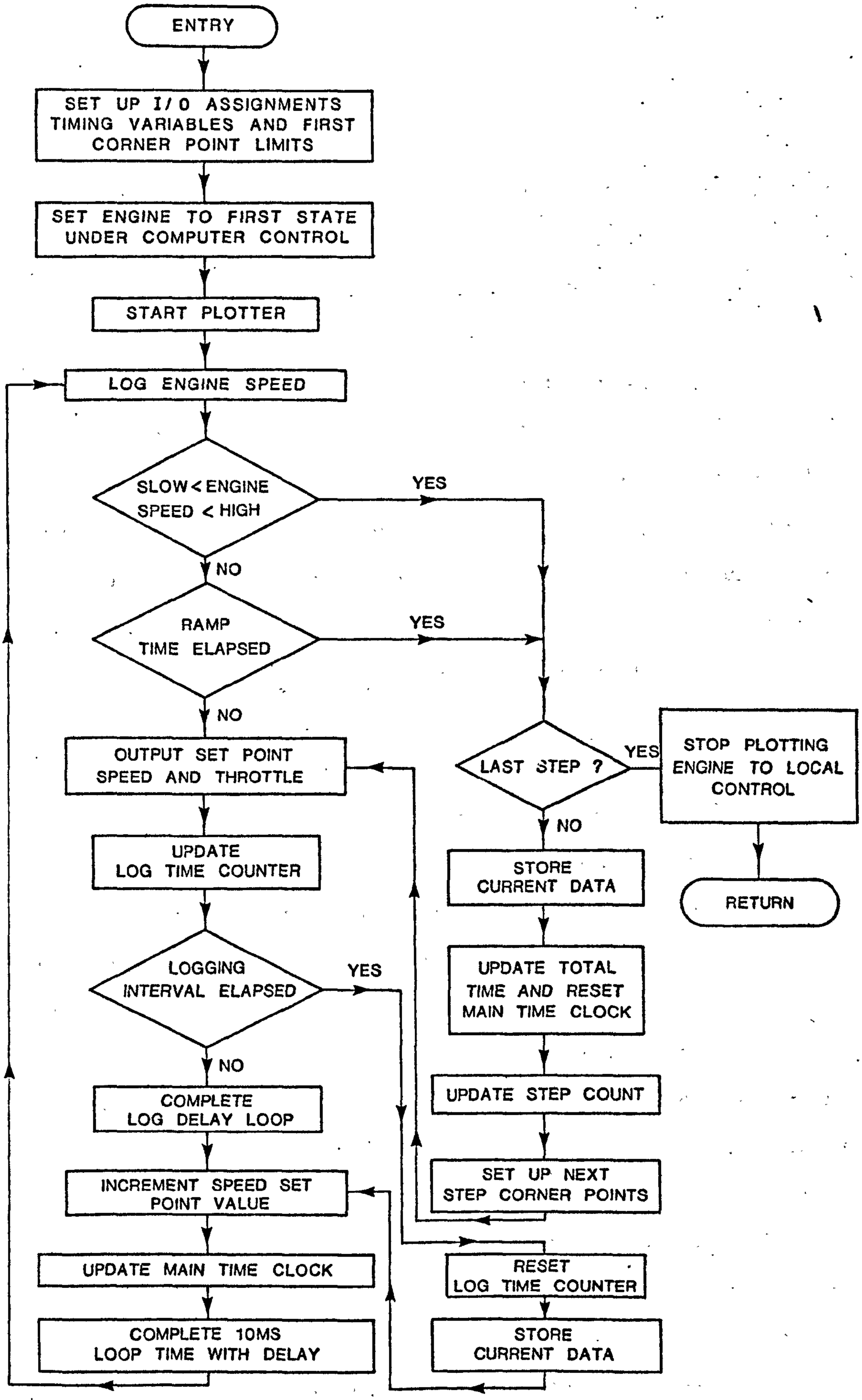


FIGURE 6.9 - RAMP ALGORITHM

clocks and set up its current pointers for engine speed, throttle position, etc. to those of the dummy first step. These idle condition engine settings are outputted to the test rig and the rig is then placed under computer control.

As well as the data logging action of the computer, engine speed was monitored by connection of the electrical speed signal from the test rig to a graph plotter. The x-axis variable was provided by using the plotter's own internal time scale. The computer was responsible for initiating this plot by pulsing a digital output connected to the plotter. By using the computer to perform this action at a particular instant fixed by the software, the plots of successive tests could be directly overlaid on the same piece of paper to provide a comparison and an indication of repeatability.

If we turn to the section of the routine concerned with actual running of the cycle we can see that our basic element is a 10 millisecond loop that uses software delays to maintain its timing. Each time around the loop the current engine speed is logged and a check is made for the occurrence of a 'corner-point' as the change over between steps is called. This can be on the grounds of the current speed being within the end of ramp speed range specified or due to an overrun of the time allowed for the step. If neither condition has occurred then the routine proceeds to update a clock that keeps track of the time elapsed since the last data storage occurred. This new value is compared with the interval calculated by the subroutine STEST. If the values are equal the clock is reset and the following data parameters are transferred to the storage area:-

i) The current time elapsed since the start of the test

ii) The current set-point for speed

iii) The latest value of speed logged

iv) The current set-point for the throttle

To maintain the validity of the overall time schedule it is vital that the loop is always 10 milliseconds in duration. Because of this, when the data storage does not occur, the routine enters a software delay of 0.35 milliseconds which is of the same duration as the processing time of the section of the program that performs the data storage.

After the data storage or delay, the pre-calculated increment in speed set point is added to current value and the time clock updated. The routine then enters a software delay that allows the remainder of the 10 millisecond interval to pass before returning to the start of the loop. This delay was determined experimentally and was found to be 6.309 milliseconds. This in fact means that the program only usefully utilises about a third of the total processor time available.

When a corner-point is detected the computer first checks to see if this is the end of the last step and hence the end of the cycle. If this is the situation, the plotter is halted and the computer relinquishes control of the test rig prior to returning to the mainline program. On other occasions a data store operation is forced, then the various timing and step counters modified. Lastly the next set of data parameters for the new step are accessed as the current values and then the next step ramp begins.

When the test cycle execution has been completed the operator already has the direct plot of speed against time performed during the cycle. It is also possible to have the computer retrieve that data stored during the test and output it on the teletype or on paper tape to allow detailed examination of the results. Finally the operator makes the choice between repeating the same test or returning to the data parameter input phase to define a new test cycle.

6.3.2 - Smoke Cycle Results

A series of test runs was performed using the software package. The modification of having a final speed demand of 2800 rev/min was used as can be seen from Figure 6.10 which is a direct plot of the voltage output from the computer digital to analog output for speed demand. It was found that a successful cycle could be performed within the limits specified but that very precise adjustment of the various gains, etc. of the local controllers was required. Furthermore, settings which one day would prove to be the optimum would have to be altered on another occasion to obtain the proper response. The reasons for this variation are difficult to isolate. Tests showed that the overall transfer function of the engine and dynamometer altered little with time although such effects as variation in ambient conditions must have a certain amount of effect. It was also felt that the analog nature of the controllers and associated signal conditioning and level switching circuits would result in their being prone to drift and calibration errors over a period of time, and that this could be partially the cause of the variation.

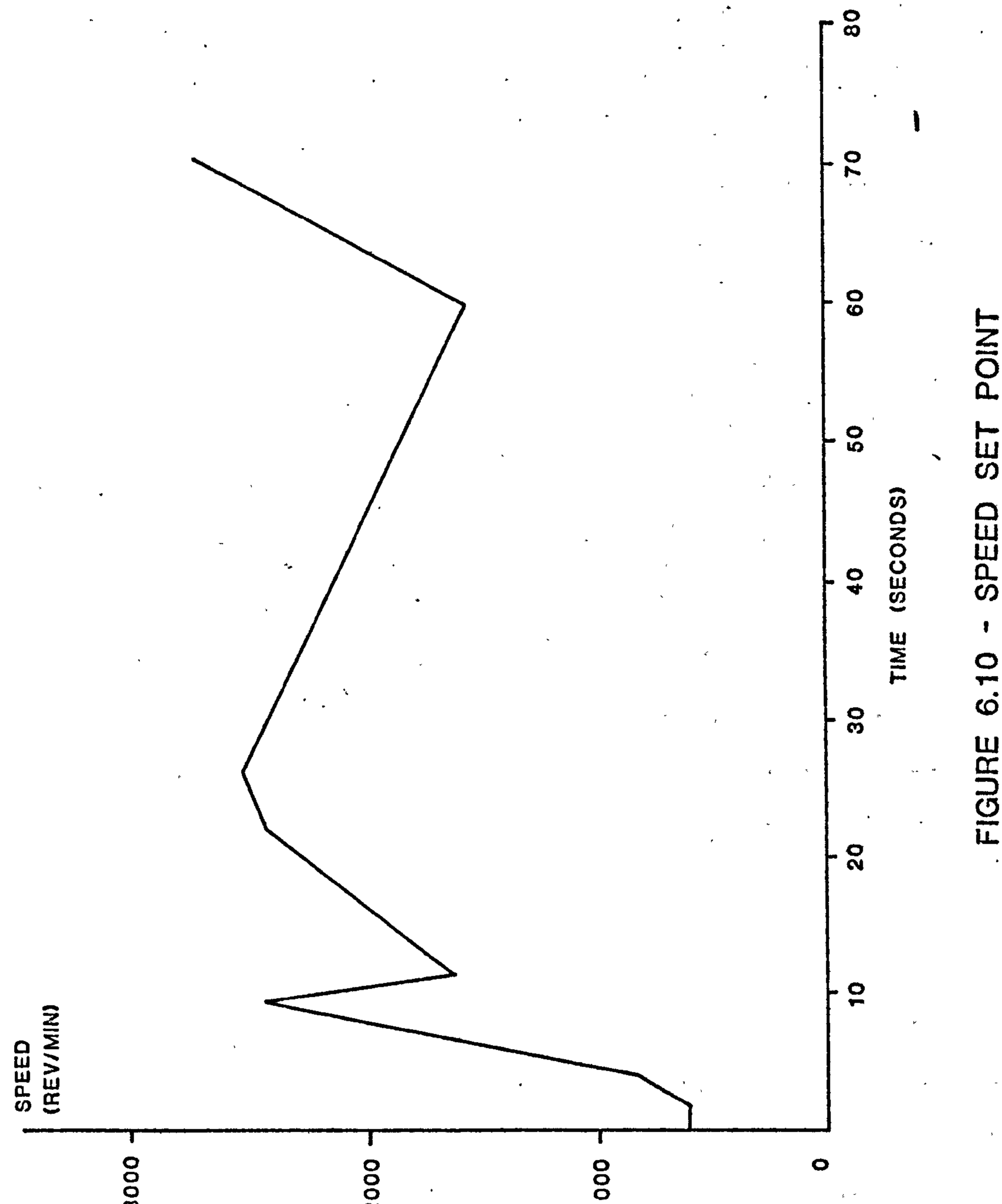

SPEED
(REV/MIN)
3000
2000
1000
0
10
20
30
40
50
60
70
80
TIME (SECONDS)
- SPEED SET POINT

FIGURE 6.10 - S

Nevertheless, given optimum adjustment of the controller settings, the system proved quite capable of performing accurate and repeatable smoke cycles. The plot shown in Figure 6.11 is engine speed against time with the speed signal being 'raw', that is the voltage signal direct from the test bed without any computer processing, and the time axis being derived from the plotters internal time scale circuit.

Although primarily developed with the U.S.E.P.A. smoke cycle in mind, the package is of course quite capable of running other tests of a dynamic nature which require the ability to set the throttle and to perform a series of controlled speed ramps. Basically all that is necessary to achieve this is a different set of input data to describe the new curve, although in some cases it might prove necessary to make adjustments to the local controller settings to achieve the desired combination of speed of response and stability for the particular disturbances to which the system is subjected.

6.4 Steady State Speed and Torque Cycles

It was decided that the ability of the system to perform steady state speed and torque cycles should also be demonstrated. The type of tests under consideration would be typically represented by the U.S. 13 Mode Cycle. The actual test procedure packages are not difficult to generate given the applications subroutine library described in Chapter 4. The only difference lies in the mode of control to be used as the control variables are speed and torque. This means

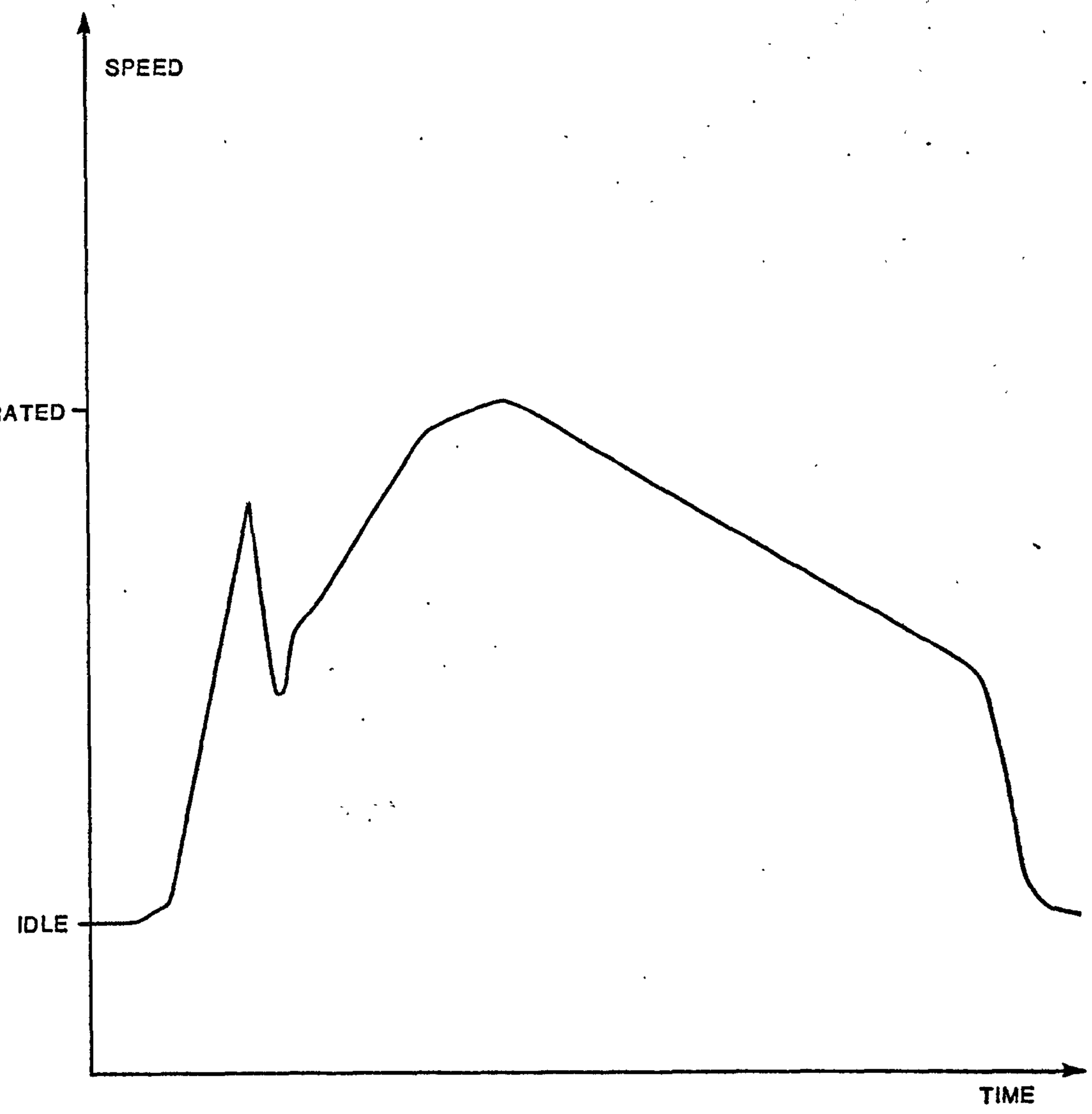


FIGURE 6.11 - SPEED SIGNAL

that we must use either control Modes I or II instead of Mode III. As far as the package was concerned, which of the two to be used was of little relevance as the only difference it made was in the setting of the digital mode control outputs. The speed and torque set points are placed on the same output lines regardless of which mode is used.

The package consisted of two main parts. The already existing package for Mode III dynamic cycles and a new section to perform steady state speed and torque cycles.

The algorithm of the new package is shown in Figure 6.12. If the Mode III type test is requested, the smoke cycle package as shown in Figure 6.6 is used except that if a repeat cycle is not required, control returns to the new master program.

When a constant speed and torque test is requested, the test title is typed and the input/output initialised. The next step is to call the subroutine STEST2 which is responsible for inputting the test set point data. This routine behaves in exactly the same way as its counterpart STEST (see Figure 6.8) except that the input data types and format differ. Figure 6.13 is the teletype listing from a 13 mode cycle except that each of the modes has been shortened to 18 seconds duration (instead of 3 minutes). When a satisfactory set of data has been inputted, the computer will check to see if the engine is ready and when it is, will begin the test. The actual running of the test cycle is performed by a subroutine called CYCLE2 (see Figure 6.14). The overall structure and behaviour of the routine is very much similar to its equivalent program, RAMP, (see

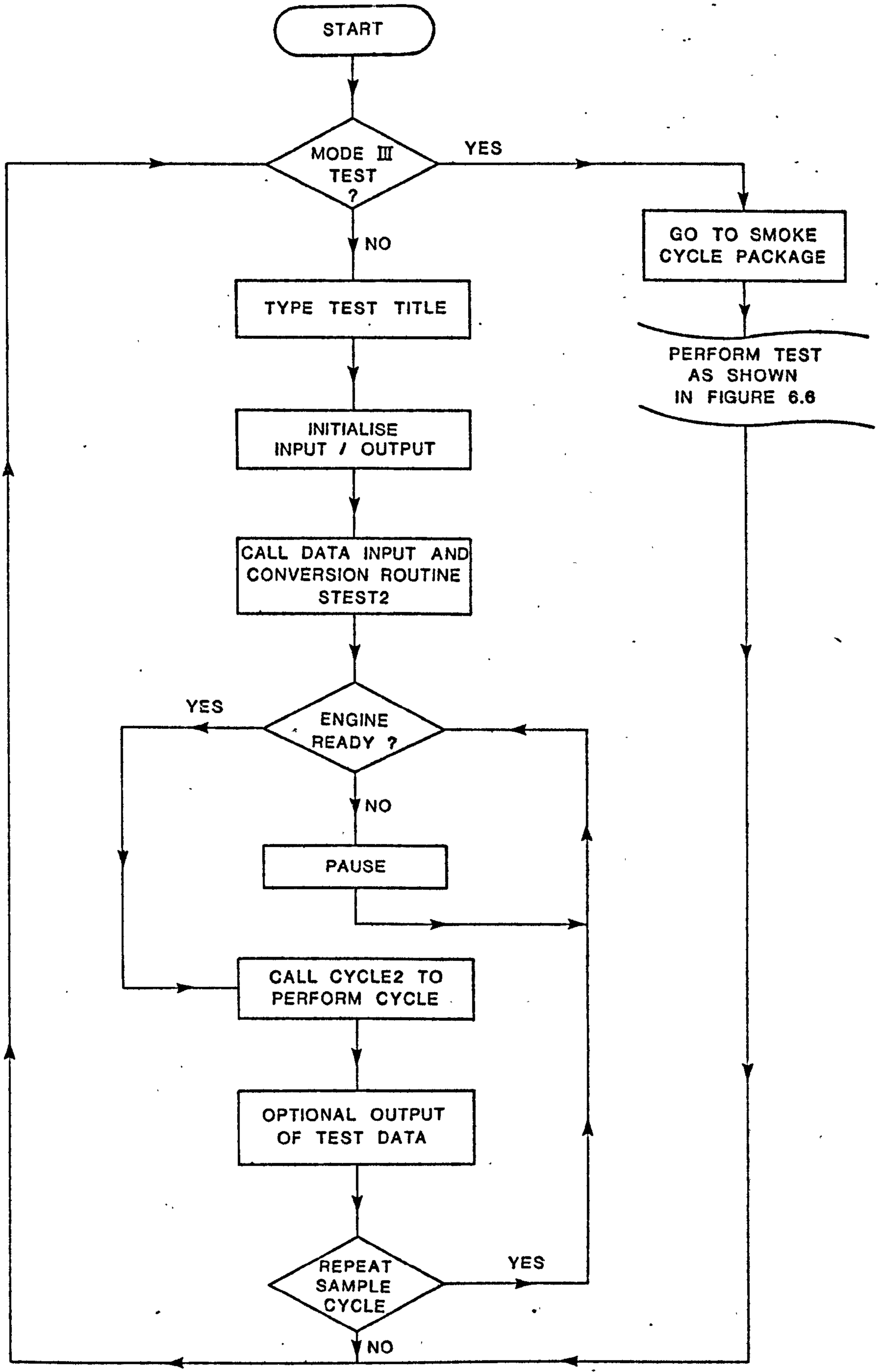


FIGURE 6.12 - CONSTANT SPEED AND TORQUE ALGORITHM

```
  ENTER REQUIRED MODE NO.
  (MODE ZERO HALTS PROGRAM)

? 1
  TYPE IN TEST TITLE
? 13 MODE CYCLE


   FEED DATA TAPE
   PAUSE 00
?

  VERIFICATION OF INPUT DATA
  NO. OF STEPS = 13

   SPEED          TORQUE          TIME
    RPM            NM             SEC
     700            0.0           18.0
    1700            0.0           18.0
    1700           70.0           18.0
    1700          140.0           18.0
    1700          210.0           18.0
    1700          280.0           18.0
     700            0.0           18.0
    2800          280.0           18.0
    2800          210.0           18.0
    2800          140.0           18.0
    2800           70.0           18.0
    2800            0.0           18.0
     700            0.0           18.0


    O.K. ?
?YES
      ENGINE READY ?
?NO
      ENGINE READY ?
?YES
      RESULTS ?
?NO
      END OF TEST
```

FIGURE 6.13 - 13 MODE DIALOGUE

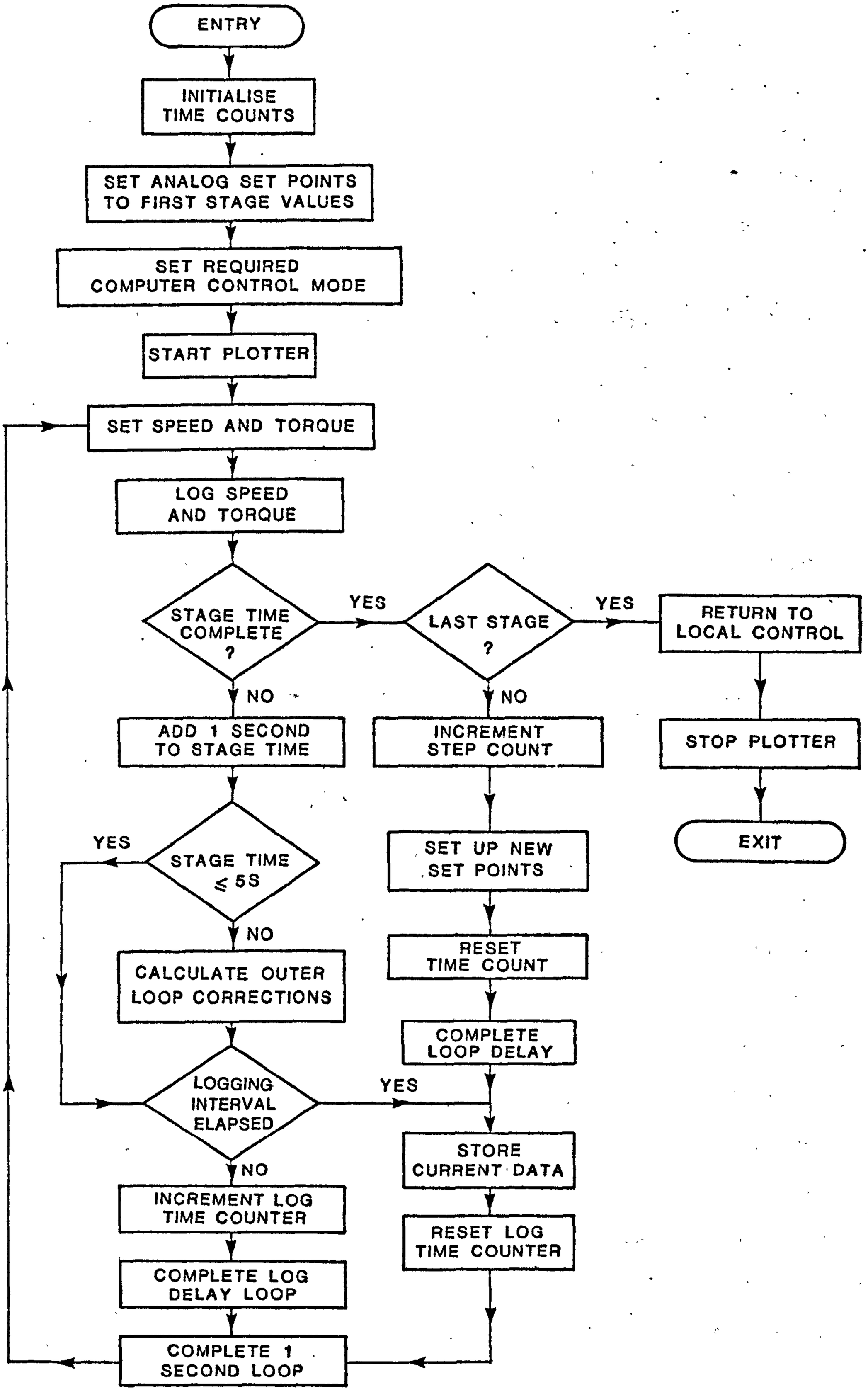


FIGURE 6.14 - CYCLE2 ALGORITHM

Figure 6.9) in the Mode III package. Obviously it is not necessary to perform incremental calculations for the set-point speed and the main control loop has been set to run only once a second. One interesting point to notice is the outer digital loops which correct the speed and torque. This part of the algorithm also shows the by-pass part which suppresses the outer loops for 5 seconds after the beginning of a new stage (or mode). This allows the engine to settle in its new steady-state conditions.

The working of the package can be seen in Figure 6.15 which shows the speed and torque set-point output signals, with the outer digital loops temporarily disabled for the duration of the test.

Besides the use of the digital loops to remove the effect of offset in the local control circuits, this method of testing relies on the test rig being able to follow the demands made on it in terms of accuracy of control of speed and torque to meet the requirements of the test cycle limits. In view of the fact that the test rig being used was capable of the much more stringent requirements of the dynamic Mode III work, it was felt that this should not be a problem. Despite this, trouble arose when attempting to run steady-state speed and torque cycles in Mode I. The system tended to be unstable. Part of the problem was due to interaction between the two control loops. The particular difficulty in using this control mode with a diesel engine is that the throttle position is in fact the governor speed lever which is in itself a speed control device. Thus there was often a case where action was being taken by two different controllers.

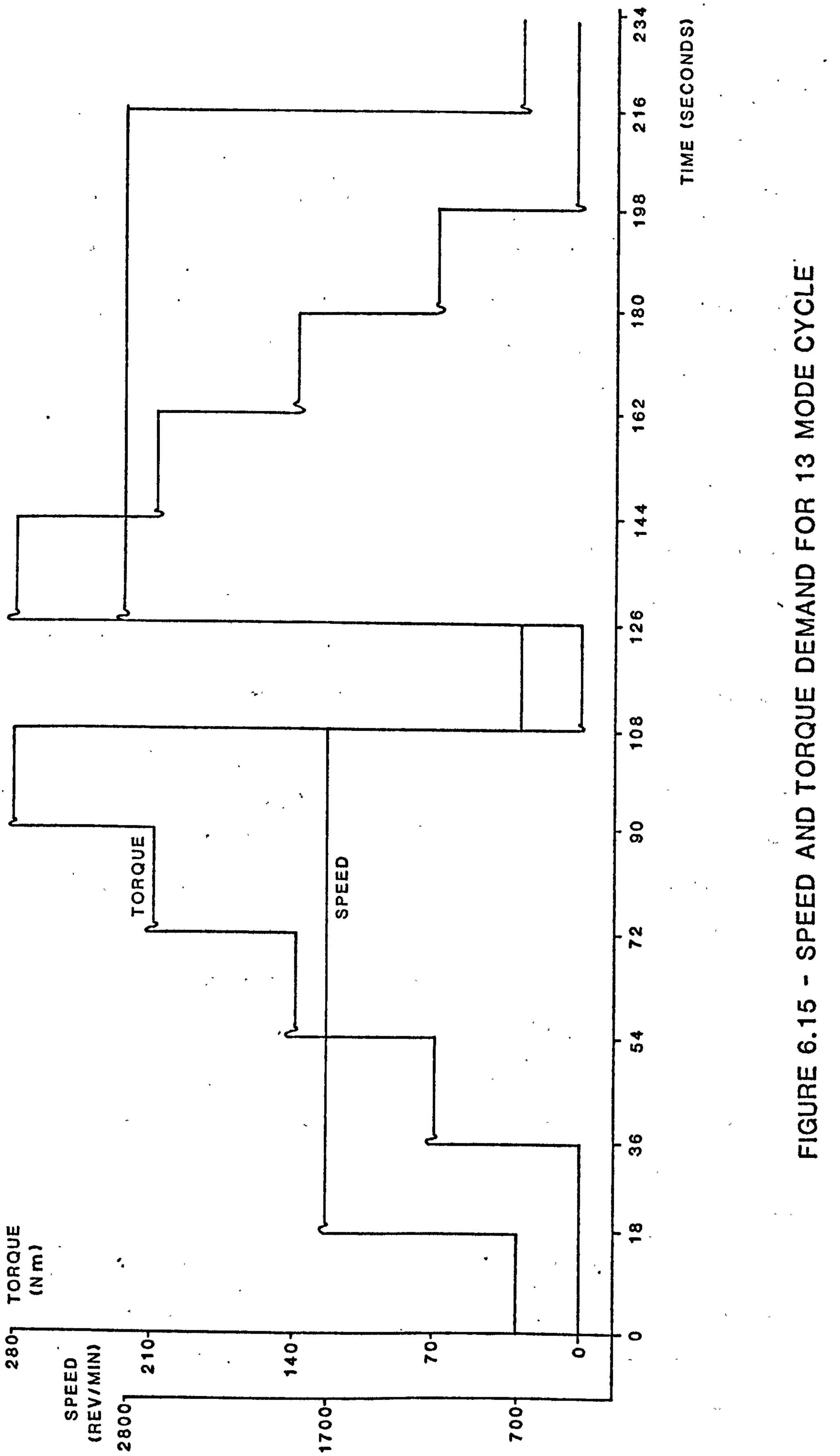


FIGURE 6.15 - SPEED AND TORQUE DEMAND FOR 13 MODE CYCLE

The difficulty was avoided by using Mode II control. The dynamometer loading provided fast accurate control of torque which is the variable that undergoes the greatest number of changes during the cycle. Using the throttle position to control engine speed worked well, although the speed of response was comparitively slow.

6.5 U.S.E.P.A. Smoke Cycle Development

Earlier in the chapter a fairly simple system for running the E.P.A. Smoke Cycle was described. The system, however, was limited in its usefullness in that it required frequent and accurate adjustment of the local controllers. Whilst this may only be regarded as a slight nuisance in a laboratory environment, it completely eliminates the practical usefulness of the system in so far as the automotive industry is concerned. With this in mind, while the development of the dynamic power curve was underway at the Ford Motor Company's Dunton Research and Engineering Centre (see Chapter 5.4), it was decided to investigate the possibility of making fuller use of the computer abilities to off-set any shortcomings of the local controllers.

It was decided that no adjustment whatsoever should be made to the existing settings of the controllers. The first stage of the investigation therefore was to examine the type of response we could expect from the local test cell system. An attempt to run a cycle in the manner used at Queen Mary College, i.e. with no outer speed control loop, was fairly disasterous, as can be seen from the speed trace of Figures 6.17 when compared to the required cycle for the

engine as shown in Figure 6.16. The next step was to run a series of tests to give some data on the actual response. By requesting various step changes in speed and monitoring the engine behaviour, it became apparent that the controllers were set for a slow but stable response which may have been ideal for steady state testing, but posed problems as far as dynamic work was concerned. As well as the slow response of the dynamometer and controller there was also a 0.5 second lag inherent in the speed control by dynamometer load.

6.5.1 - Adaptive Learning

The approach decided upon to improve the computer controlling abilities, was that of adaptive learning. This means that the computer performs a test run of the particular cycle under consideration and then modifies its set point output for the next test run, on the basis of the response obtained. As there is also the effect of time lags to be taken into account, this adaption should also take into consideration what the response in the future will be, as well as what is happening at any particular instant. To illustrate how this is done let us consider three arrays, X, Y and Z. Their dimension is time and their magnitudes represent engine speeds:

X = Desired speed (as set out by the definition of the cycle)

Y = Computer set point as output at any given time during the last cycle.

and Z = Monitored Engine Speed.

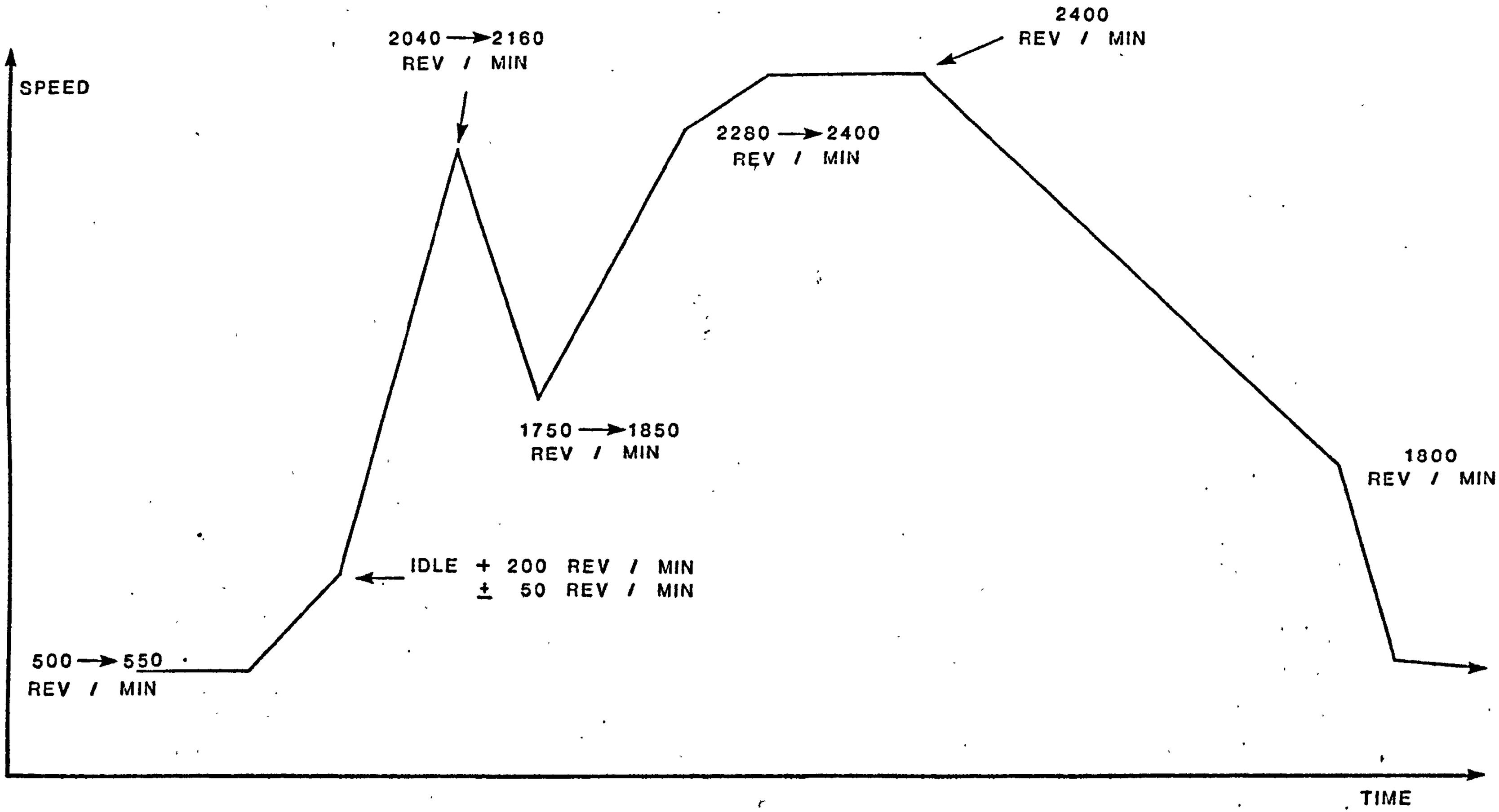


FIGURE 6.16 - SMOKE CYCLE FOR TORNADO ENGINE

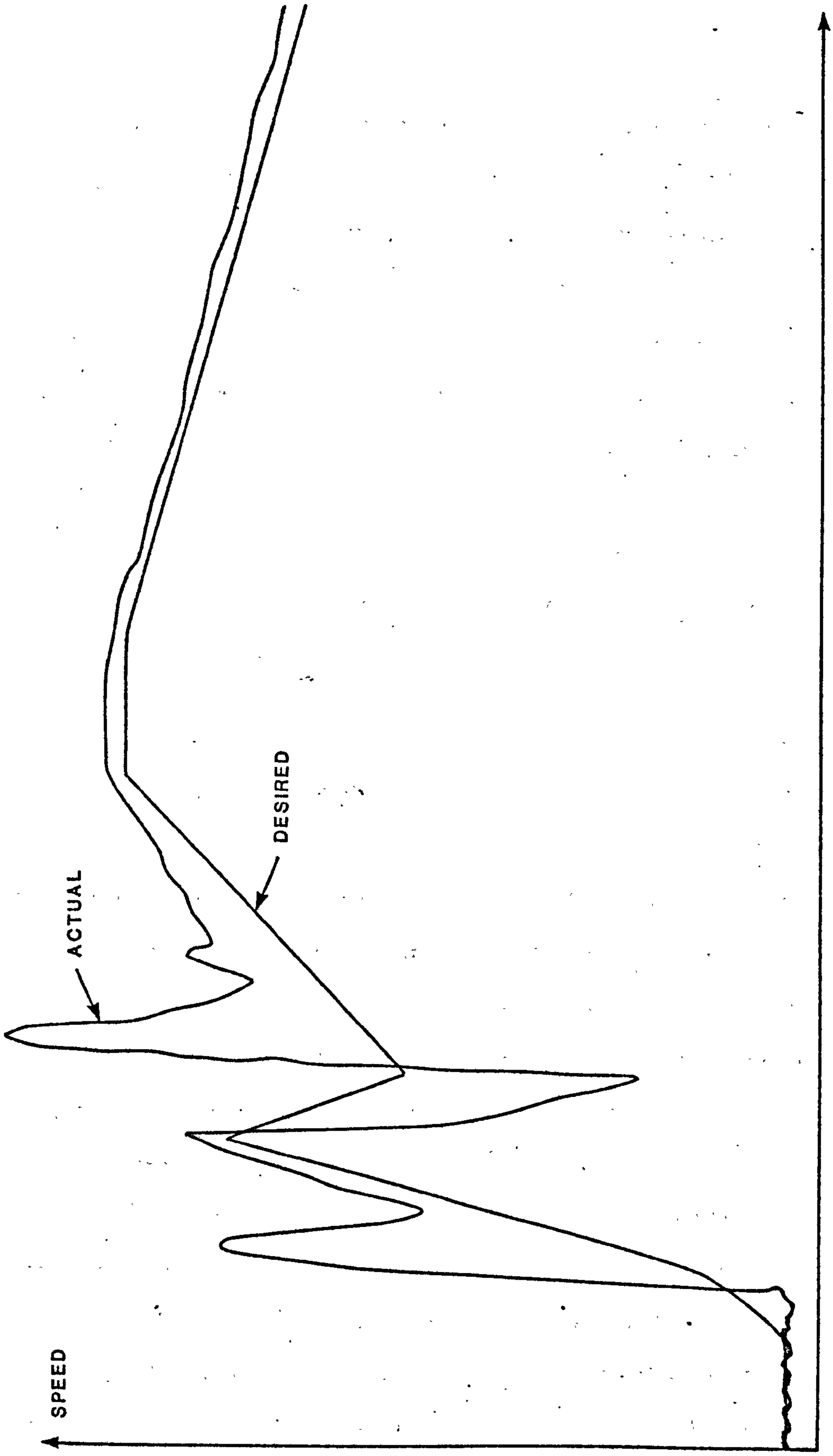


FIGURE 6.17 - INITIAL SPEED RESPONSE

If all the arrays are arranged to be orientated from $X_o \rightarrow X_n$, $Y_o \rightarrow Y_n$ and $Z_o \rightarrow Z_n$ during the course of the cycle, then the performance of the system at any instant in the test can be described by X_r, Y_r and Z_r. We now define a new array $A_o \rightarrow A_n$ which will be the computer set point output for our next test run and will hence become the Y array after that test. This new array provides the adaptive ability and is calculated using the following equation:-

$$A_r = Y_r + ((X_r - Z_r) \times C_1) + ((X_{r+k_1} - Z_{r+k_1}) \times C_2) \qquad (6.1)$$

where C_1 and C_2 are constants and k_1 is a time offset so that the time point in an array given by $r + k$ occurs some time after the point signified by r. The value of k should be adapted to match the lag of the system and C_1 and C_2 form weighting factors which allow the reaction to current errors and trends to be controlled. This adaptive method is also shown diagrammatically in Figure 6.18.

Although this method formed the basis of the technique, it was felt that some additional ideas should be incorporated so as to allow the corner-points, i.e. the changeover between stages of the cycle, to be treated as special cases. Previous experience had already shown that these points were the major locations for inaccuracies. This is only to be expected if one considers the situation. During the progress of a smooth ramp there is little disturbance to the system and hence the controller, even if badly adjusted, should progressively

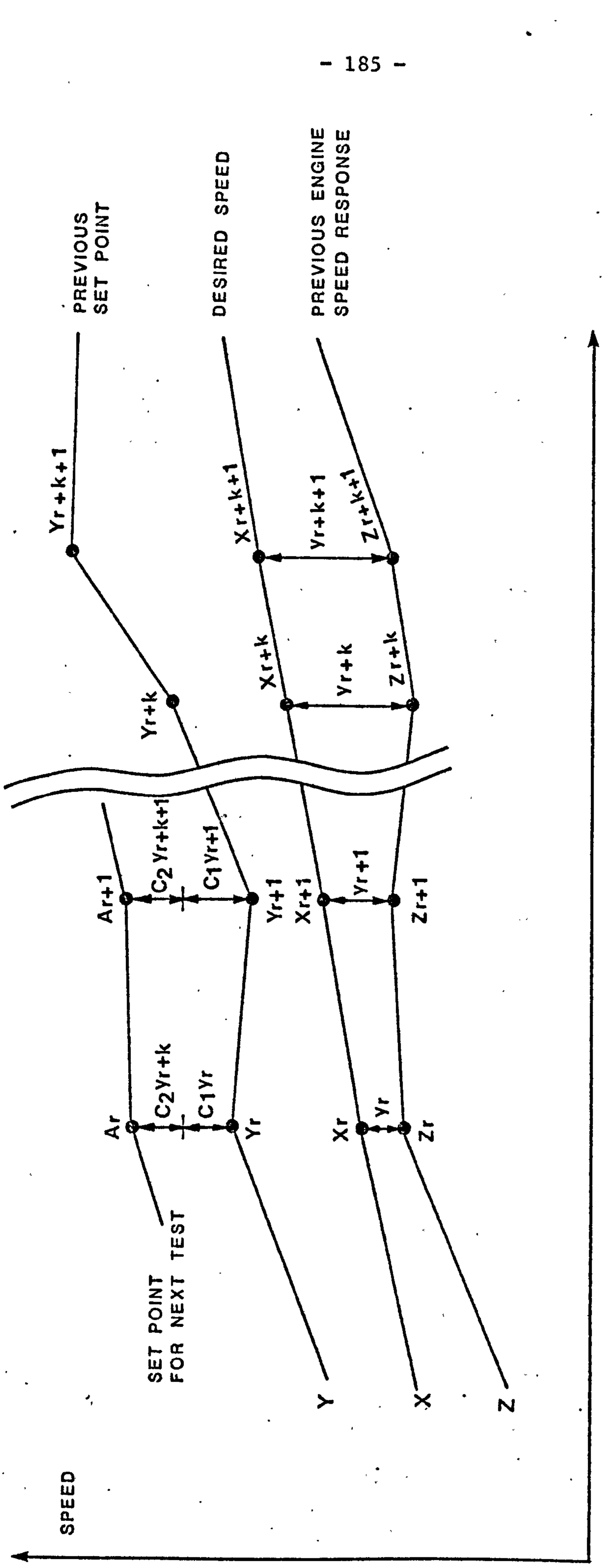


FIGURE 6.18 - ADAPTIVE METHOD

reduce the error. However, at the corner point, a major disturbance occurs in the speed demand and, to make matters worse, at several of the corner points in the smoke cycle this is also associated with a disturbance caused by a major alteration in the throttle position.

To improve the corner point response, it was decided that during set point correction, corner points should be separated and a more powerful forward looking correction factor applied. The modified equation is:-

$$A_r = Y_r + ((X_r - Z_r) \times C_1) + ((X_{r+k_1} - Z_{r+k_1}) \times C_3)$$
$$+ ((X_{r+k_2} - Z_{r+k_2}) \times C_4) + ((X_{r+k_3} - Z_{r+k_3}) \times C_5) \qquad (6.2)$$

for $k_3 > k_2 > k_1 > 0$

It should be noted that the weighting factor for the r term is C_1 for both equation 6.1 and 6.2 but that the weighting applied to the $r + k_1$ term is independent, i.e. C_2 for ordinary points and C_3 for corner points.

The next stage was to write a software program that would be capable of using the techniques described above. Obviously the new package was largely based on the earlier version but with changes made to the sequencing of the set points. The new strategy is

illustrated in Figure 6.19 which demonstrates how each stage of the cycle is now split into steps of 100 milliseconds duration. It is the start and end points of these steps that are optimised using equation 6.1. In turn, these ramps are generated by a series of equal magnitude 10 millisecond steps in set point. The overall algorithm of this new package is shown in Figure 6.20. The first action of the package is to read in the list of parameters which form the definition of the cycle. The format used is much the same as in the earlier version but another parameter has to be defined for each of the cycle stages. This is called the mode and two different modes are defined in the software (see Figure 6.21). These are:-

Mode 1 - The speed is to be ramped and throttle is stepped at the start of the stage. This is the only mode used in the earlier Smoke Cycle Package.

Mode 2 - The throttle is to be ramped and the speed stepped at the start of the stage.

This second new mode of operation was added as a result of experience gained during early work on the USEPA smoke cycles. The particular part of the cycle for which the new mode was intended, was the first section when the speed is accelerated from idle to a speed 200 rev/min higher with some throttle opening. Using the original method (Mode 1), Figure 6.22 shows that almost immediately the controller is required to apply large amounts of load via the dynamometer to prevent the engine from following its normal free acceleration curve. Bearing in mind that the engine had previously been at idle

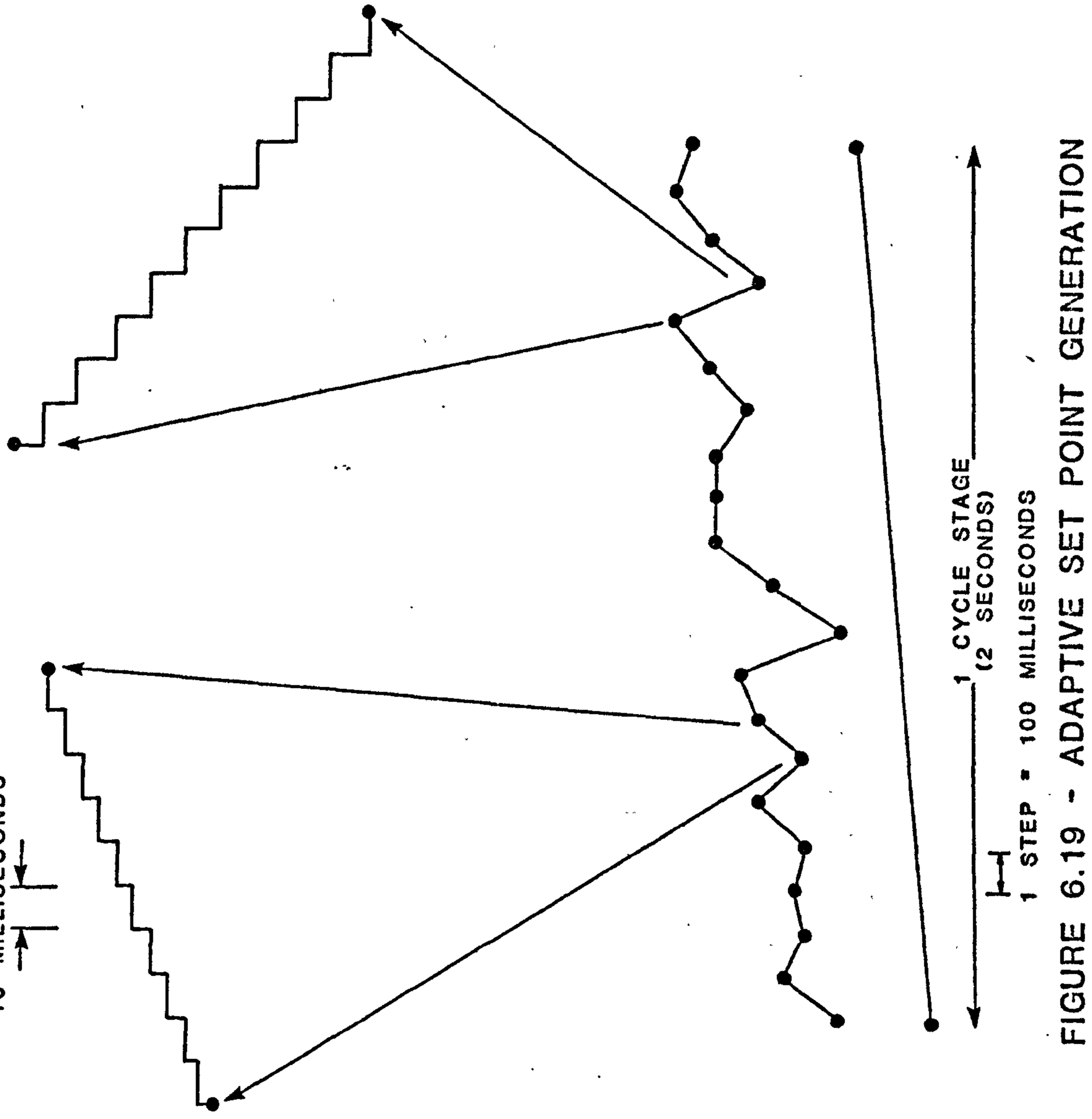


FIGURE 6.19 - ADAPTIVE SET P

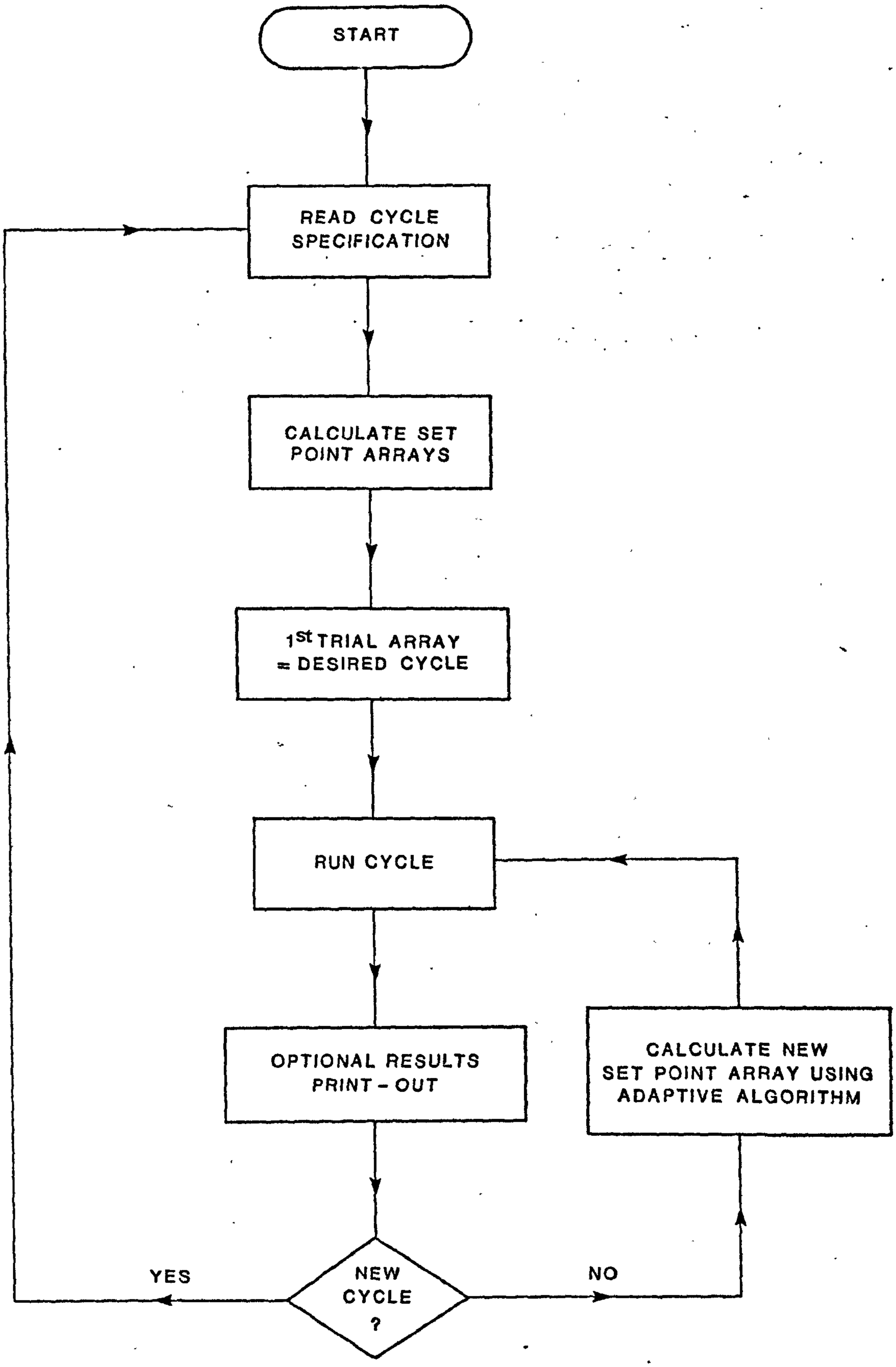
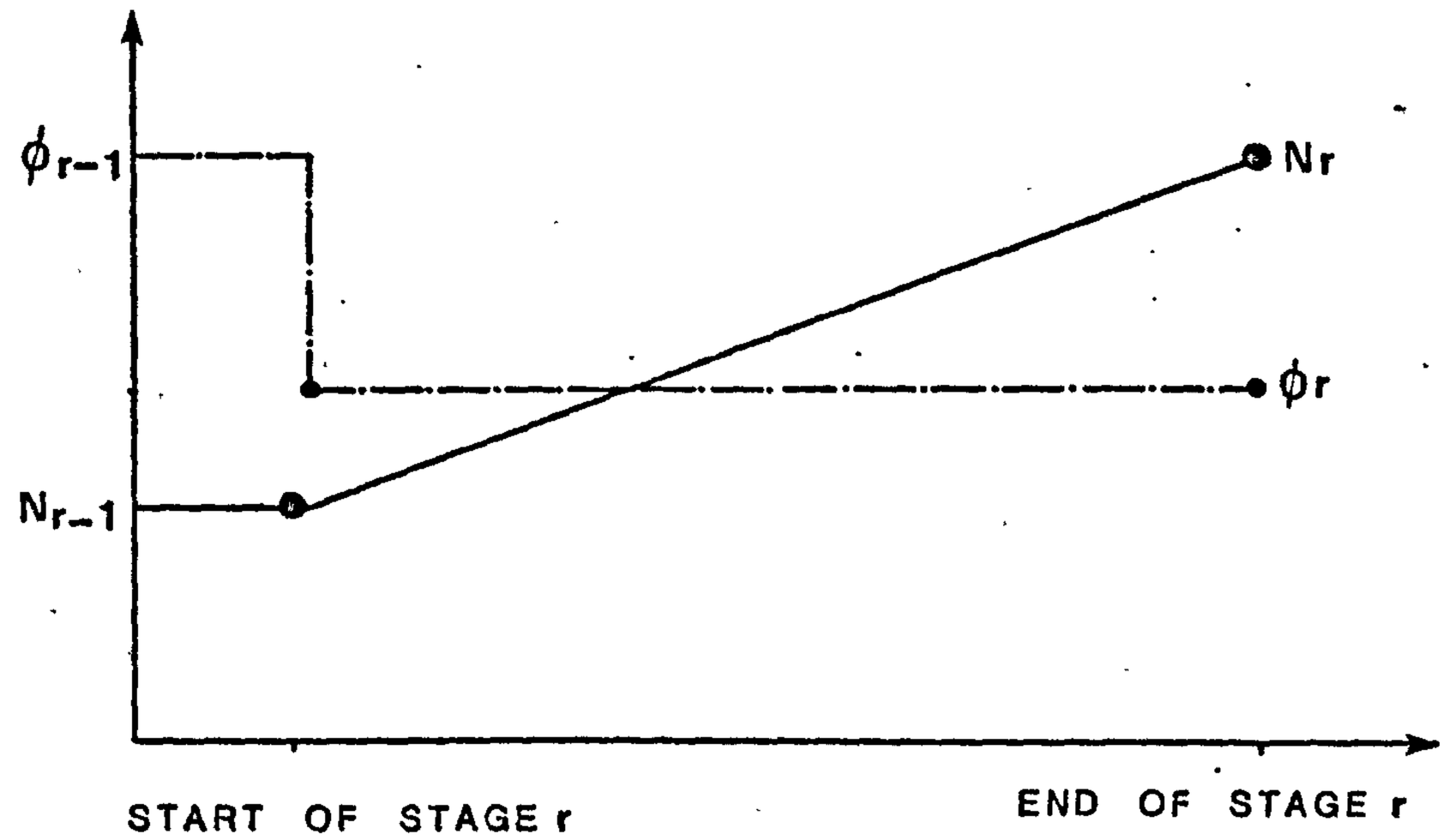


FIGURE 6.20 - ADAPTIVE PACKAGE ALGORITHM

MODE 1

SPEED FOR STAGE r = Nr
THROTTLE FOR STAGE r = ϕr



MODE 2

SPEED FOR STAGE r = Nr
THROTTLE FOR STAGE r = ϕr

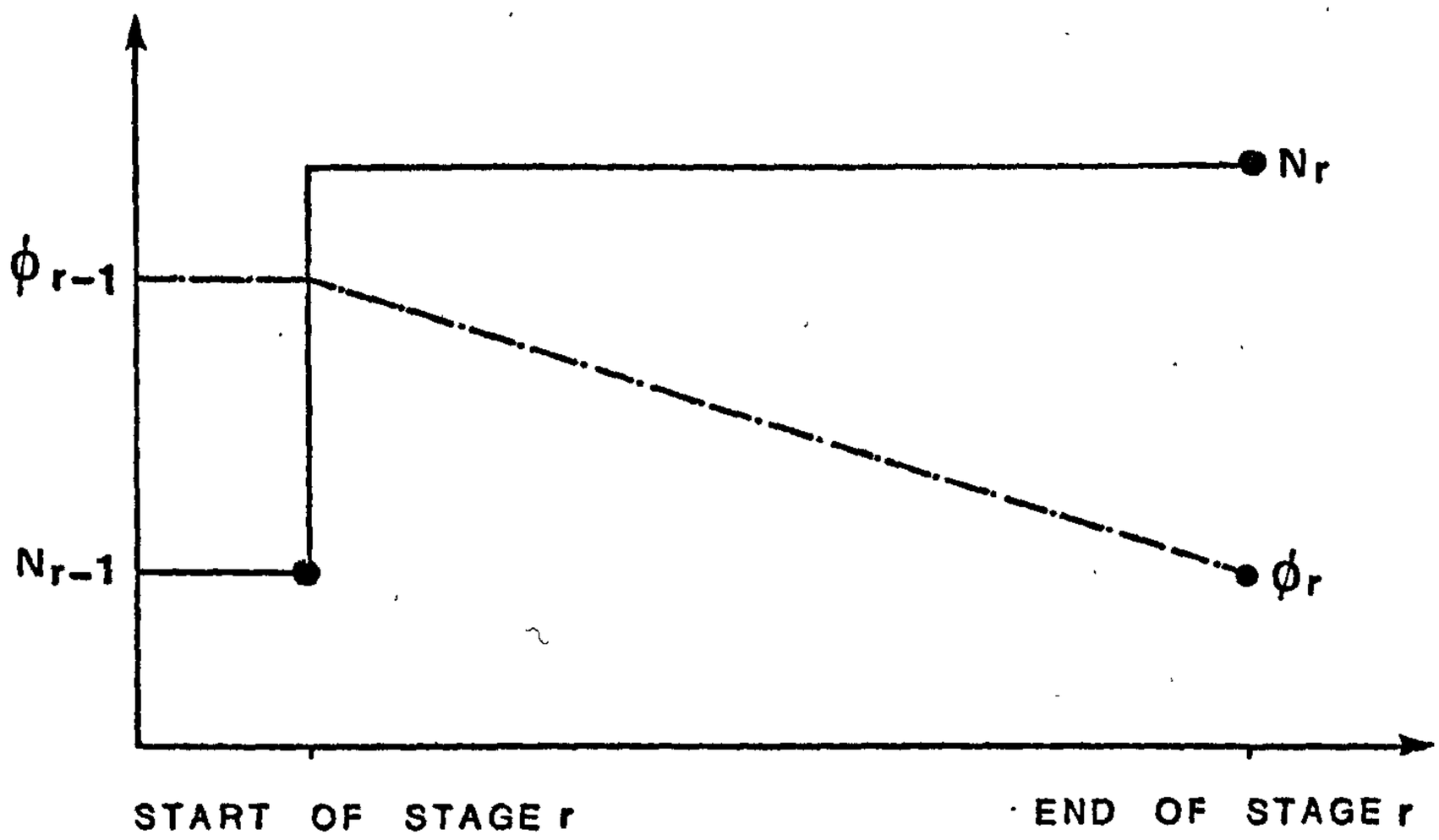


FIGURE 6.21 - CYCLE STAGE MODES

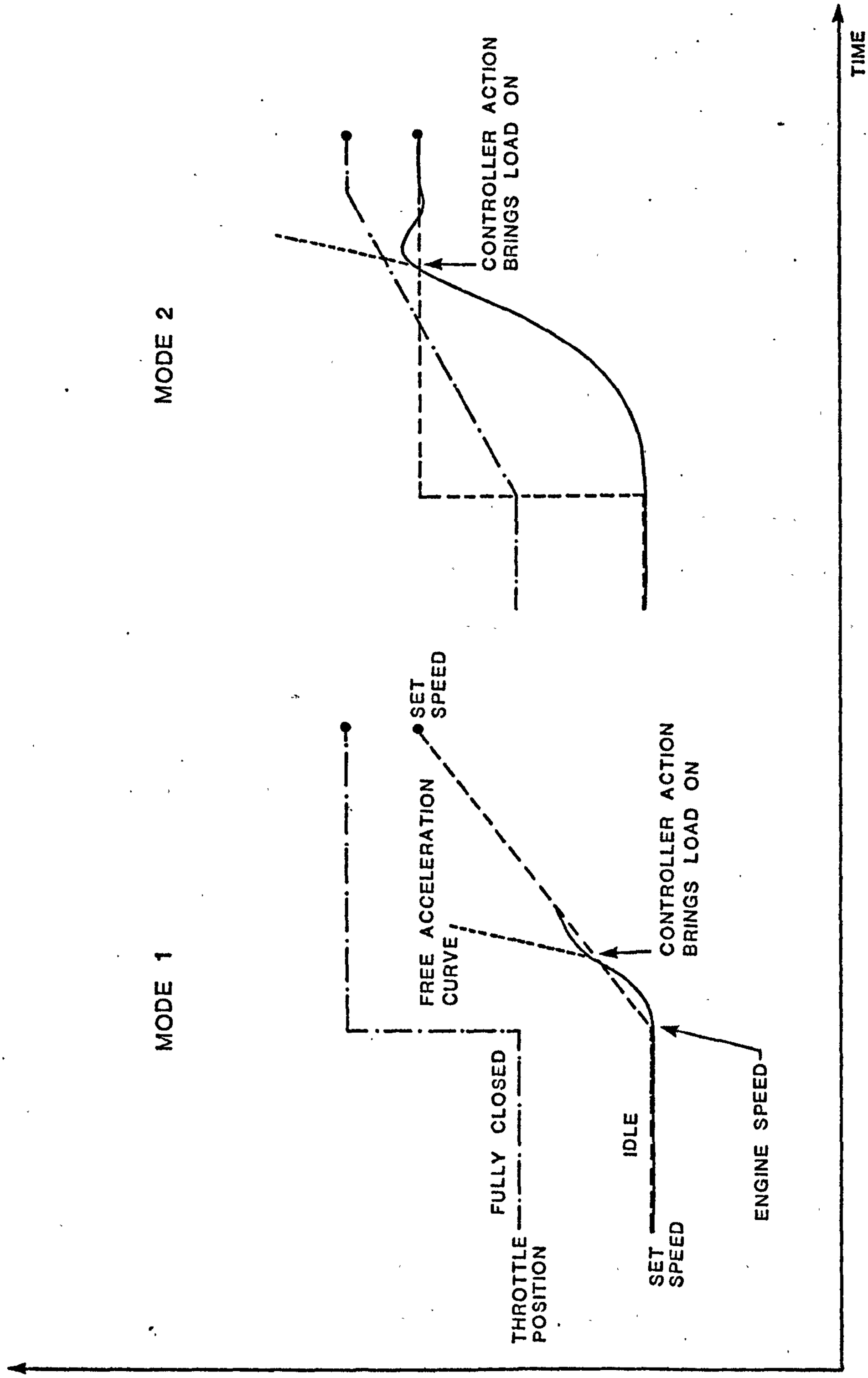


FIGURE 6.22 - CYCLE STAGE 1

for a considerable period of time, this sudden transition had been found to cause a very high peak in the exhaust smoke level. Now from the point of view of the objective of simply running an engine through the legally required cycle, the mode 1 response was adequate. In practical terms, test equipment users, particularly if they are engine manufacturers, do not want to see engines failing the test they would have passed had it not been for the initial peak smoke reading. Because of this, some thought was given to a possible solution, and the use of the second mode was planned. As can be seen from the drawing in this case the slow opening of the throttle, gives a slow acceleration curve and it is not until late in the stage that any load at all is applied. The fact that the speed no longer follows a ramp does not invalidate the test as there are no linearity constraints on this section of the cycle.

It should perhaps be mentioned that there is at first sight another, even better way, in which to avoid a high initial load. In a diesel engine, the throttle is essentially a speed controller itself, and it should be possible to open gradually the throttle to the right amount to bring the speed to the required value of idle plus 200 rev/min with no load. The problem however, even ignoring the task of defining for each engine the exact throttle opening required to give the correct speed, lies in the next stage of the cycle. At the corner point following the first stage the throttle is opened fully. However the speed ramp of the second stage must have linearity within ± 100 rev/min and as the throttle must be fully opened, there would be difficulty in meeting the linearity due to the

speed of response when starting from no load conditions.

When the computer has obtained the cycle data, it calculates an array representing the desired speed at 100 millisecond intervals throughout the test. This is used as the set point and the cycle is run with the engine response monitored. The results are then used to calculate the next array of set-points which are then used during a test followed by another adaptive calculation and so on until the operator intervenes and stops the process.

In addition to the above major alterations, a good many minor improvements were also made to the software package. These include background overspeed alarm and shutdown monitoring facilities and also the use of hardware rather than software timing control.

6.5.2. - Adaptive Results

Initially it should be remembered that our starting point was the very poor control performance shown in Figure 6.17. The first operation of the investigation was to define the values for the various constants of equations 6.1 and 6.2. The first weighting factor, C_1, is analoguous in operation to the simple digital outer loop system of Chapter 4 as it deals with current off-sets. Thus its value was set to unity.

The normal look-ahead delay, k, should ideally be equal to the speed control loop finite time lag. A series of step response tests identified this as being approximately 0.5 seconds so that k_1 was set to 5 to give a look ahead interval of five 100 millisecond steps. The values of k_2 and k_3 were set to look at the next

two points in the array, i.e. $k_2 = 6$ and $k_3 = 7$. The values assigned to the other weighting factors were:-

$$C_2 = 0.5, \quad C_3 = 0.5, \quad C_4 = 0.25 \quad \text{and} \quad C_5 = 0.1$$

Test runs using both the full cycle and simplified versions were then tried. It was found that the technique of adaptive learning did improve the cycle, particularly during the second half when the throttle opening is constant (see Figure 6.23), but the response when throttle position alteration caused cut-in or cut-out of the turbocharger was still poor after a large number of runs. It was felt that a considerable amount of work would be needed to improve the behaviour of the correction algorithm. Due to limits on the time available, it was decided instead to limit the investigation to showing that the outer loop technique could perform adequate correction of the controller action even though the setting up the modifed points would be done on a manual rather than automatic basis. This was done on a more or less trial and error basis. A series of runs, each examining a specific section of the cycle were performed. Examples of these can be seen in Figures 6.24 and 6.25 where the objective is to obtain the required linearity of the second stage of the cycle. The software package has been modified to allow simultaneous ramps of speed and throttle position. The result of this work can be seen in Figure 6.26 which shows the engine following the desired cycle quite closely, thus demonstrating the overall feasibility of outer digital loop control. It should be remembered that the

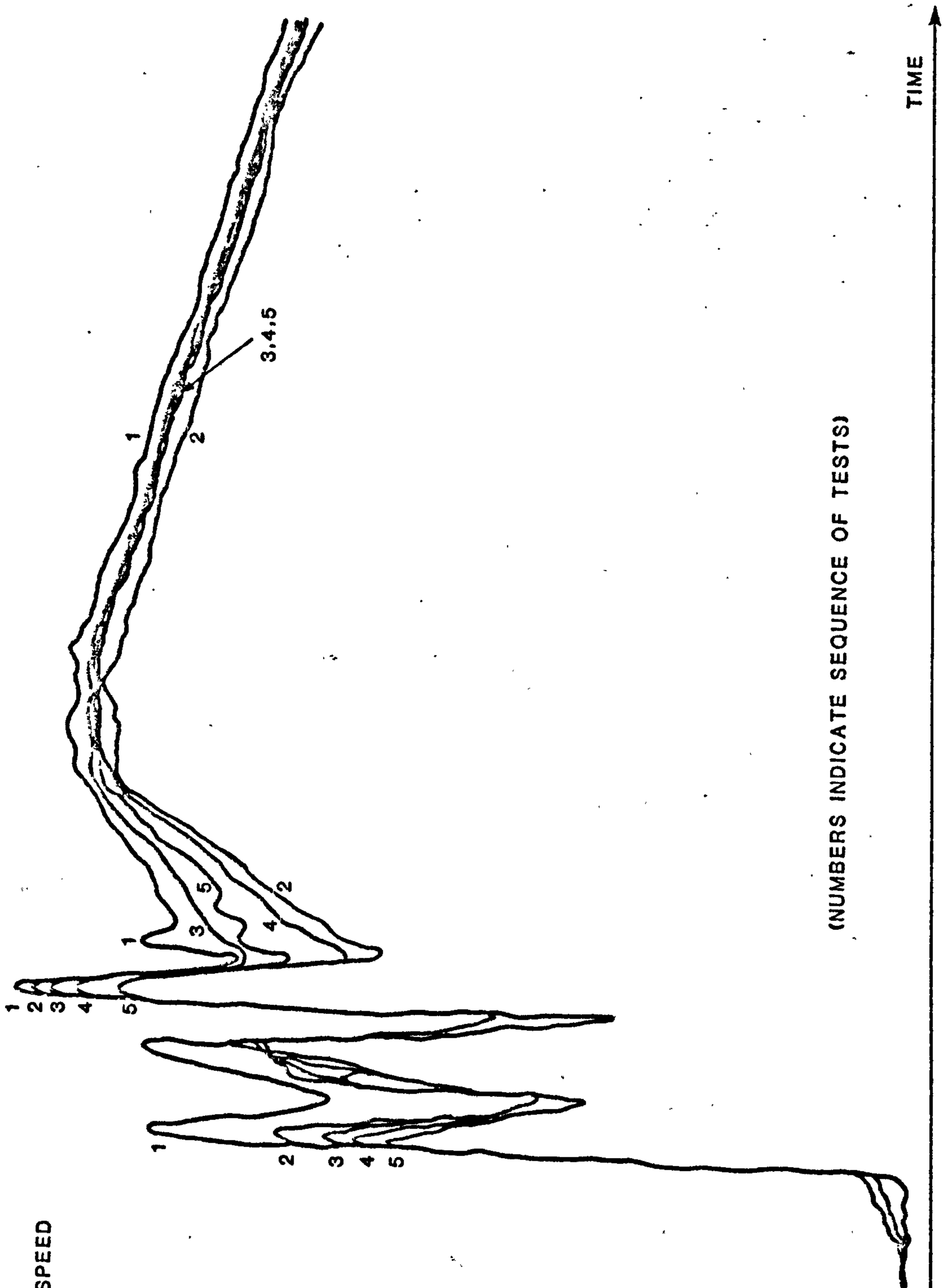


FIGURE 6.23 - ADAPTIVE CORRECTION

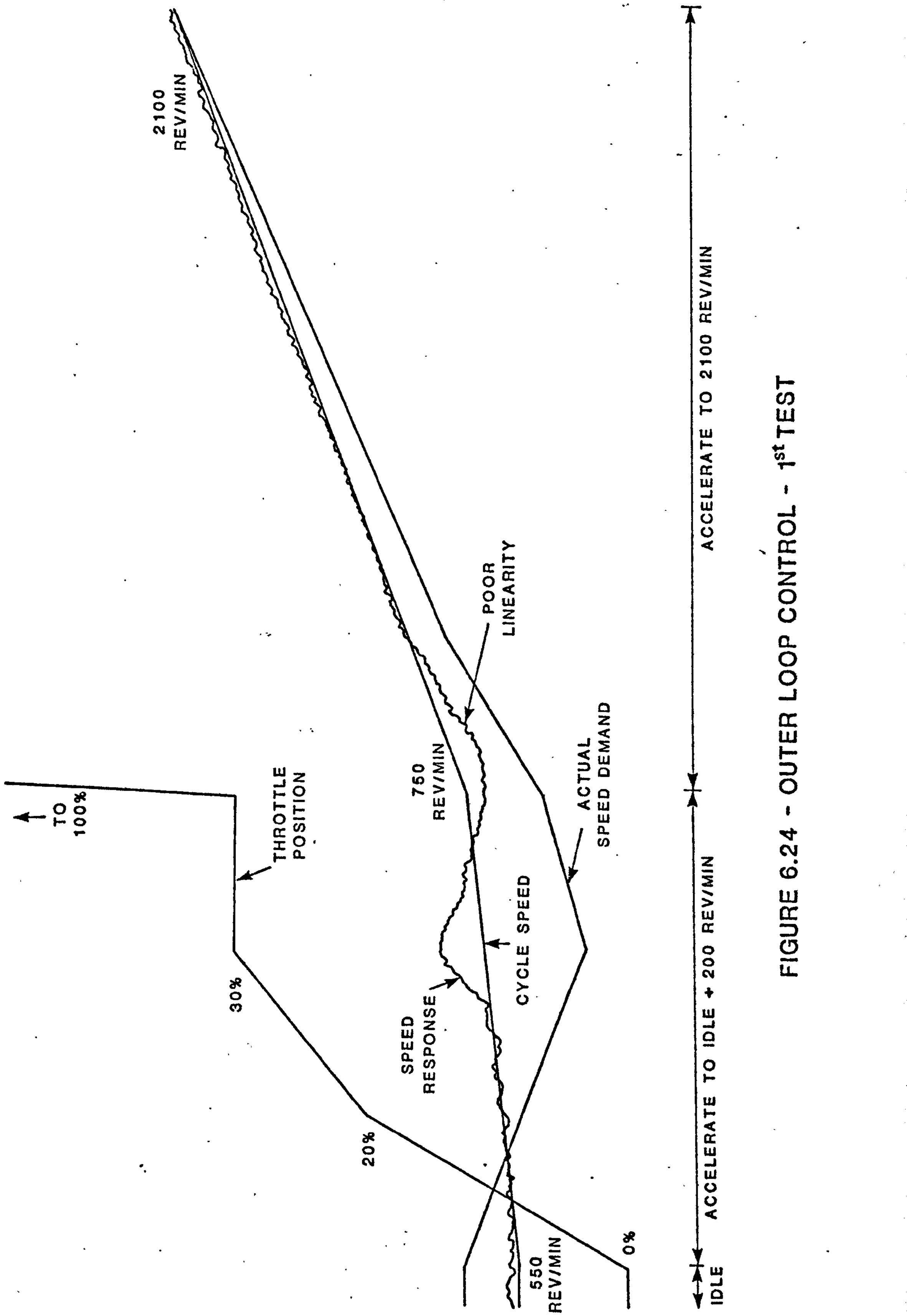


FIGURE 6.24 - OUTER LOOP CONTROL - 1st TEST

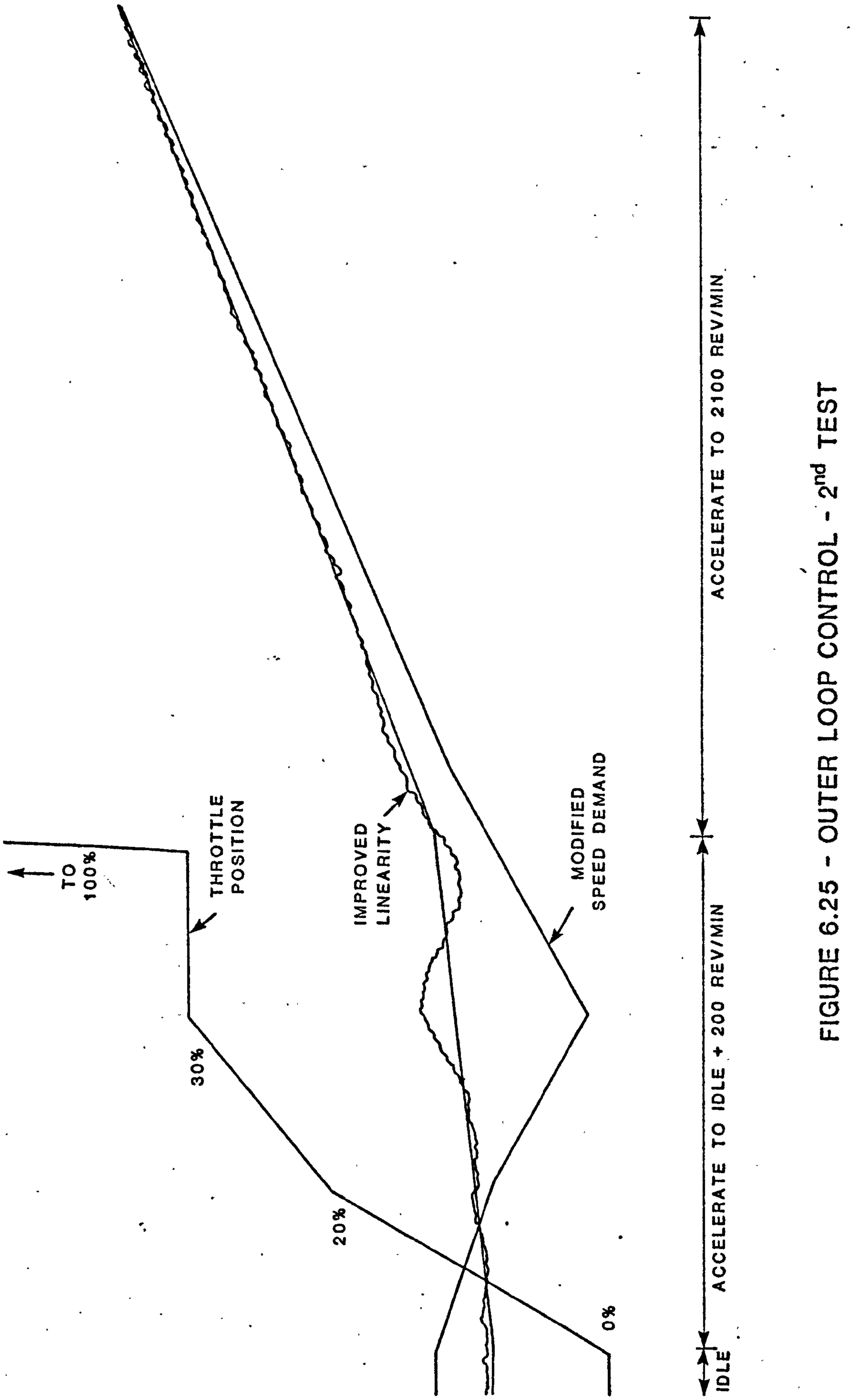


FIGURE 6.25 - OUTER LOOP CONTROL - 2nd TEST

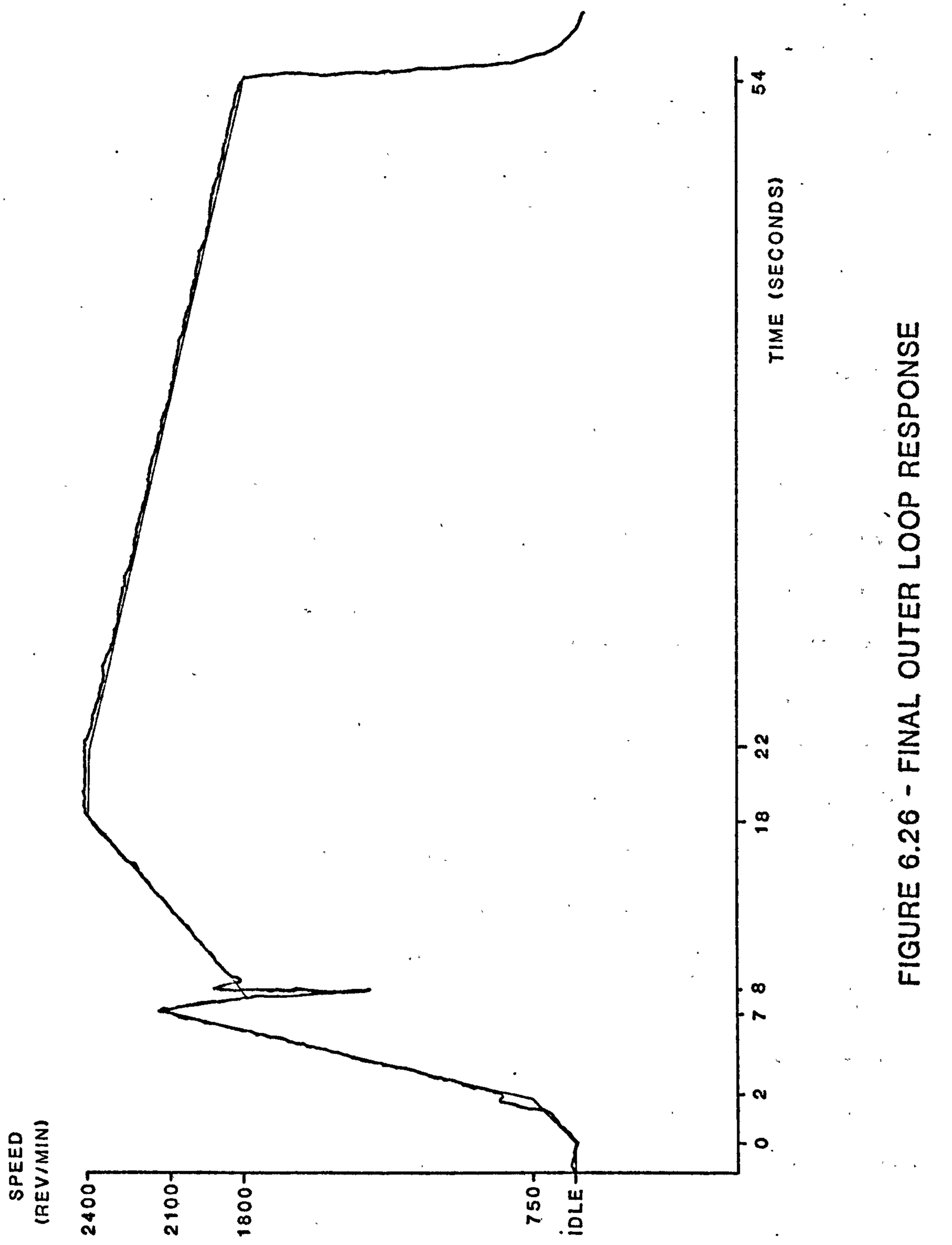


FIGURE 6.26 - FINAL OUTER LOOP RESPONSE

local controller gain settings were not altered throughout the testing and so the results of Figure 6.26 should be seen against a background of a system comprising of a difficult test, a turbocharged engine and controllers with poor dynamic response.

6.6 Summary

In order to provide realistic data on engine exhaust emissions, it is often considered necessary to use dynamic test cycles. To allow the legal requirements on emissions to be properly enforced, it is vital that a high degree of uniformity is achieved when testing engines to identify their emission levels. This means that an engine test system used for this purpose must be able to perform dynamic cycles with very high accuracy. It has been shown that, taking the USEPA Smoke Cycle as an example, the legally defined test cycle can be performed using the minicomputer only to output the basic demand values for speed and torque. This, however, requires that the local analog controllers should be adjusted to give the optimum dynamic response. An alternative approach would be to make more use of the computer ability to perform additional controlling operations in the form of outer loop processing. Although further work is needed to implement a fully automatic correction system, the feasibility of using set point modification has been demonstrated and an approach to the oyerall problem put forward.

Chapter 7

ENGINE AND TESTING SIMULATION

7.1 Introduction

At some points in other chapters mention is made of various simulations of engines and test systems. As in many cases the same simulation, or parts of it, have a wide range of usage, the different simulation exercises have all been concentrated into one chapter, with the appropriate reference being made where required.

The use of simulation techniques is particularly valuable in the field of developing automated process control systems. One of the prime advantages is the fact that one does not require access to the actual process equipment during the development phase. This fact has several important ramifications. In the case where the process equipment itself is also being developed, it is not necessary to await its completion before performing experimental work with the process control equipment, thus allowing the development work to proceed in parallel and hence speeding up completion of the work.

Even should the process equipment already be in existence, the use of simulation techniques is still important. It is often, for example, difficult to justify economically tying up an expensive engine test cell for a long period whilst the automation system is under development and when the test cell could instead be performing other useful work. Additionally there is a greater safety margin provided by the initial use of a simulated process. It is a virtually unavoidable fact that early in their development lives most automation systems will show misbehaviour due to errors in the software. In a simulation environment such a failure of control may

only result in, for instance, the advent of an overload warning light of an analog computer rather than the occurrence of damage or destruction of an expensive engine in the case of the real process.

Another advantage to the use of simulation is the improvement in flexibility which can result. In a simulation, the simple alteration of a numerical value can represent a large physical modification. Thus, for example, the behaviour of a process control system with different engines, and/or dynamometers, can be easily investigated.

It should be emphasized that the simulation models, described in this chapter, do not necessarily represent high accuracy approximations to the real life situations. This sort of work is only useful when it is the simulated process itself that is being studied. For our purposes here it is only necessary to generate a simulation which is sufficiently true to life as to provide a reasonable test of the functioning of our process control system.

7.2 Analog versus Digital Simulation

In simulating a real process we can make use of either analog or digital electronic techniques. Each has its own characteristic advantages and disadvantages. Although digital processing can provide facilities such as low cost hardware, high flexibility, high noise immunity and ease of programming, it is difficult to perform the real time simulation of any complex physical system using a single digital processor. This is because such a simulation will invariably involve the solving of first or second order differential

equations. Using an analog computer such solutions can be obtained directly from a representational circuit using integrators and unless we are dealing with a process which has a faster time response than our electrical circuits, there will be no problem in maintaining a real-time solution. In the case of digital processing, however, there is no direct method to solve these differential equations and usually an iterative method of solution has to be employed. This takes considerable time to execute with the result that in any moderately complex simulation, the use of digital techniques will preclude having the simulation proceed as fast as the real process.

From the above reasons it can be seen that both types of simulation have their place. For basic studies where real time performance is not important, digital techniques provide ease and flexibility. However when developing actual process control systems, particularly in a dynamic testing environment where the timing factor is critical, analog simulation is the answer.

One final point that should perhaps be made is that we have been considering digital simulation using a single processor. It would be possible to get round the processing time problem by using several processors to handle different parts of the simulation in parallel rather like the different amplifiers and other elements of an analog computer, instead of serial processing techniques. In the past this solution would not have been possible due to the high cost but the advent of the low cost microprocessor may well make such a technique attractive in the future.

7.3 Analog Test Bed Simulation

There are many examples already in existence of complete analog simulations of engine and dynamometer combinations which represent a test rig. Most of these are very complex and require the services of a large analog computer system together with a lot of setting up. When using a simulation to test a new automation package it should be remembered that it may not be necessary to have such a comprehensive simulation. The simulation need only be valid for the planned engine speed and load states that will normally be encountered during the test. As long as the simulation will also give a clear indication of when it is being asked to operate outside the limits of its validity, there is no reason why a comparatively simple system cannot be used in the early development phase.

7.3.1 - Full Load Simulation

An example of this limited simulation is the one used in connection with the development of the various full load power curve test systems described in previous chapters. As the test should be run from start to finish with the throttle fully open, it would be pointless to expend the time and effort in producing a test bed simulation which is valid for all throttle positions. Furthermore the speed/torque characteristic at constant throttle opening of an engine is usually represented by a quadratic equation, with an additional function for the case of a diesel engine operating above the governor cut-in speed. Before going to the stage of committing oneself to simulating

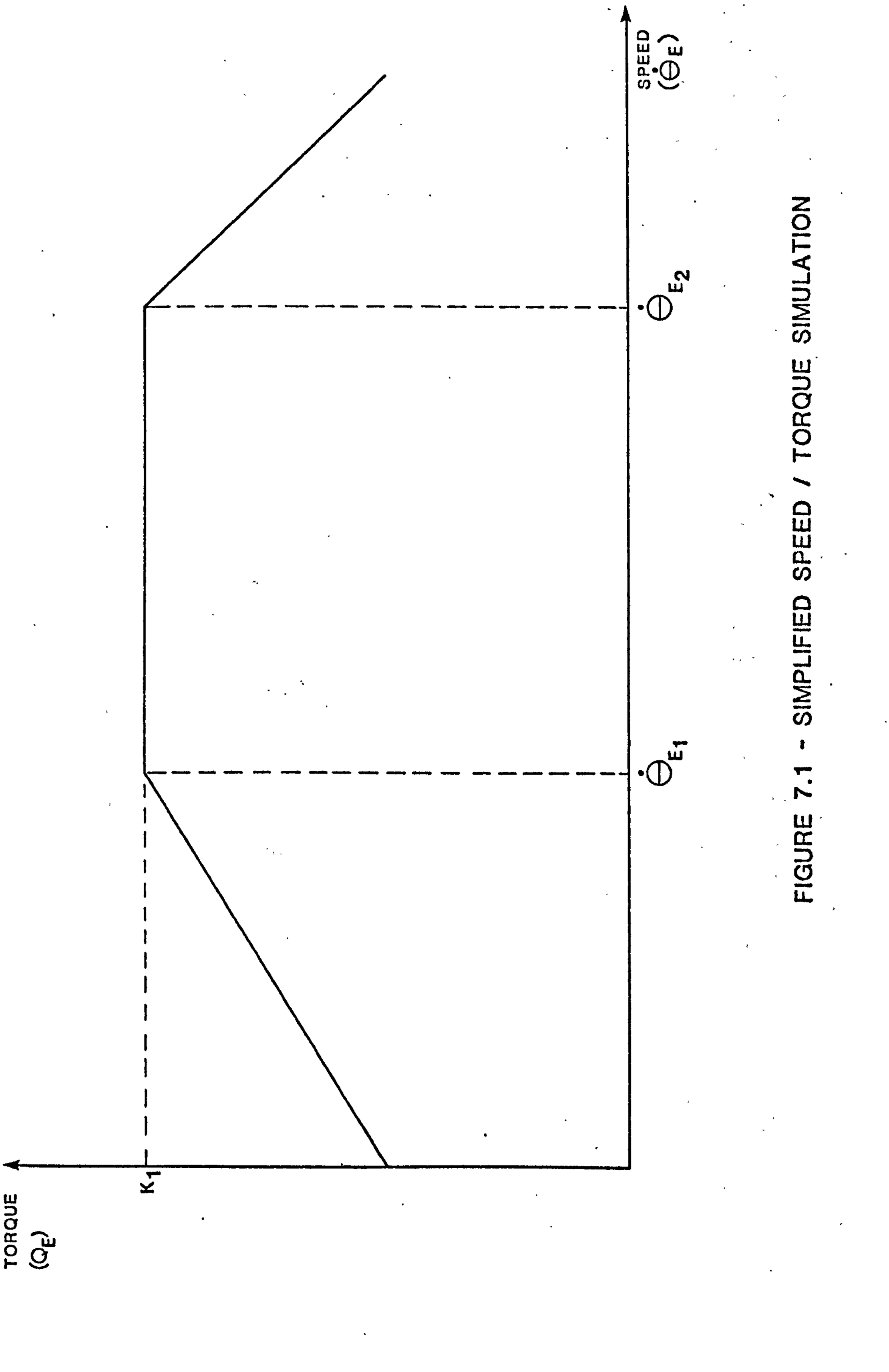


FIGURE 7.1 - SIMPLIFIED SPEED / TORQUE SIMULATION

this in its entirety, the question of whether such accurracy is necessary should be considered. If our only requirement is to test the operation of the new software package then a very crude approximation to the curve is all that is necessary.

The result of all this is that the actual simulation used is very simple. As far as the throttle opening is concerned the simulation merely states that unless the throttle is fully open then the torque output of the engine is zero, i.e.:-

$$Q_E = 0 \quad \text{for} \quad \phi < 100\% \qquad (7.1)$$

Thus if any misbehaviour in the throttle set position output occurs due to a fault in this system this will immediately be shown up as the engine torque drops to zero.

The full load speed/torque relationship is again the subject of considerable simplification and is defined as consisting of three sections as shown in Figure 7.1. The simulation states that, given constants K_1, K_2, and K_3, the torque (Q_E) is given by:-

$$Q_E = K_1 - K_2(\dot{\theta}_{E_1} - \dot{\theta}_E) \qquad (7.2)$$

$$\text{for} \quad \dot{\theta}_E < \dot{\theta}_{E_1}$$

$$\text{and} \qquad Q_E = K_1 \quad \text{for} \quad \dot{\theta}_{E_1} \leq \dot{\theta}_E \leq \dot{\theta}_{E_2} \qquad (7.3)$$

$$\text{and} \qquad Q_E = K_1 - K_3(\dot{\theta}_E - \dot{\theta}_{E_2}) \qquad (7.4)$$

$$\text{for} \quad \dot{\theta}_E > \dot{\theta}_{E_2}$$

The above relationship does not take into account dynamic effects due to change of speed. Such considerations are not really necessary in connection with steady state power curves, but to show the offset in torque resulting from a dynamic ramp, a simple first order lag was added to the torque output. The engine speed was assumed to be equal to the computer demand value. The resulting simulation can be seen in Figure 7.2.. Two comparators are used to control which ever of the three equations (7.2., 7.3., and 7.4.) apply at the present speed. Whenever one of the two terms $f_1(\theta_E)$ and $f_2(\theta_E)$ is not applicable, the relevant input of the torque summing amplifier is grounded. A third comparator is used to ground the Q_E signal in the event of the throttle position dropping below 100%.

This simulation was extensively used during various parts of the power curve development work. Figure 7.3 shows the computer output after running a full-load power curve when using the simulation.

7.3.2 - General Test Rig Simulation

To allow rather more generalised use of simulation techniques to be used in connection with developing engine test automation systems a somewhat more comprehensive simulation has been developed, based on some earlier work at Queen Mary College (6). The model of the engine is fairly accurately based on a heavy-duty diesel engine and is valid for the entire operating range of the engine. The dynamometer simulation is a simplified one.

In contrast to the previous example, where the ideal local controllers can be assumed to be incorporated in the simulation (as

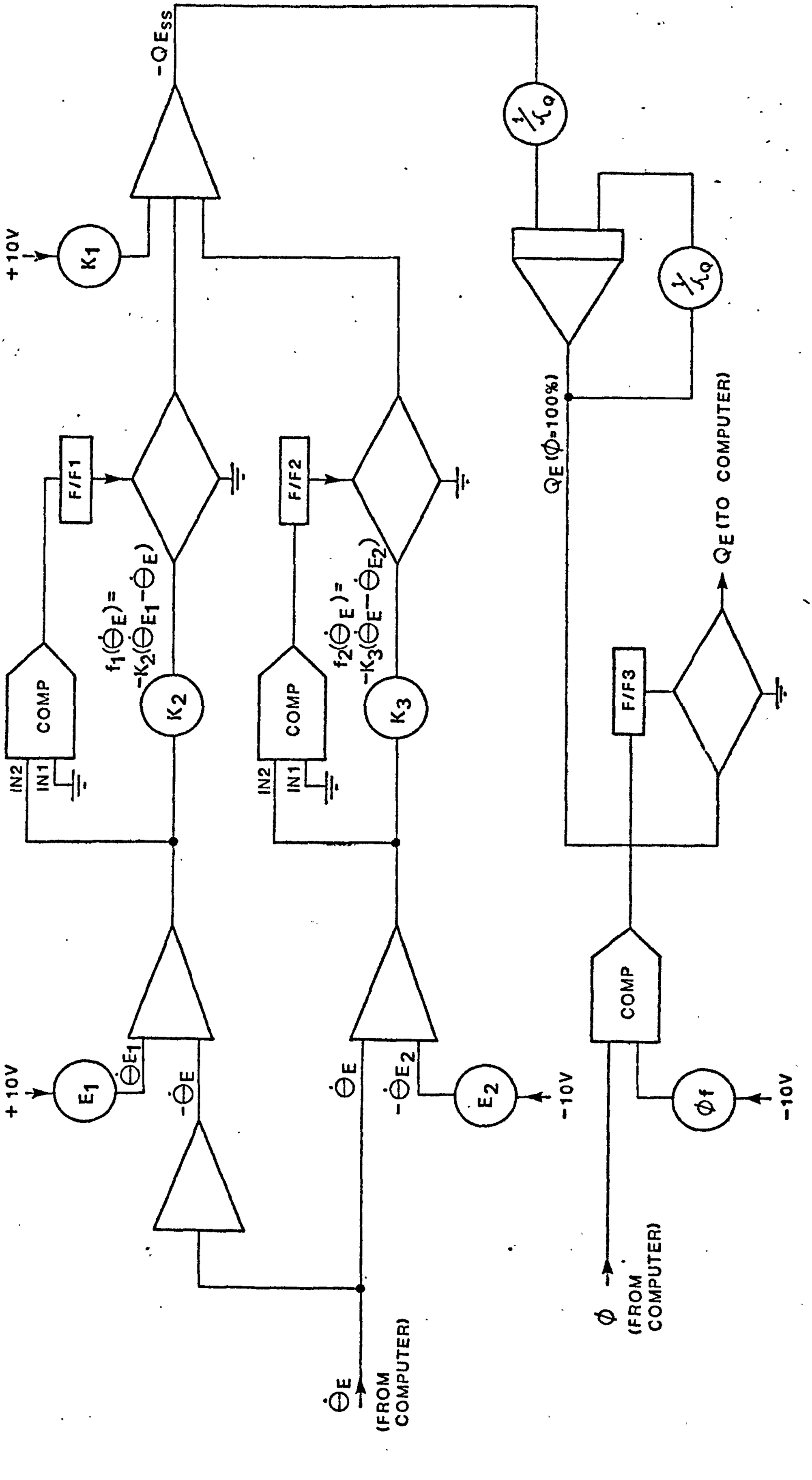


FIGURE 7.2 - FULL LOAD SIMULATION

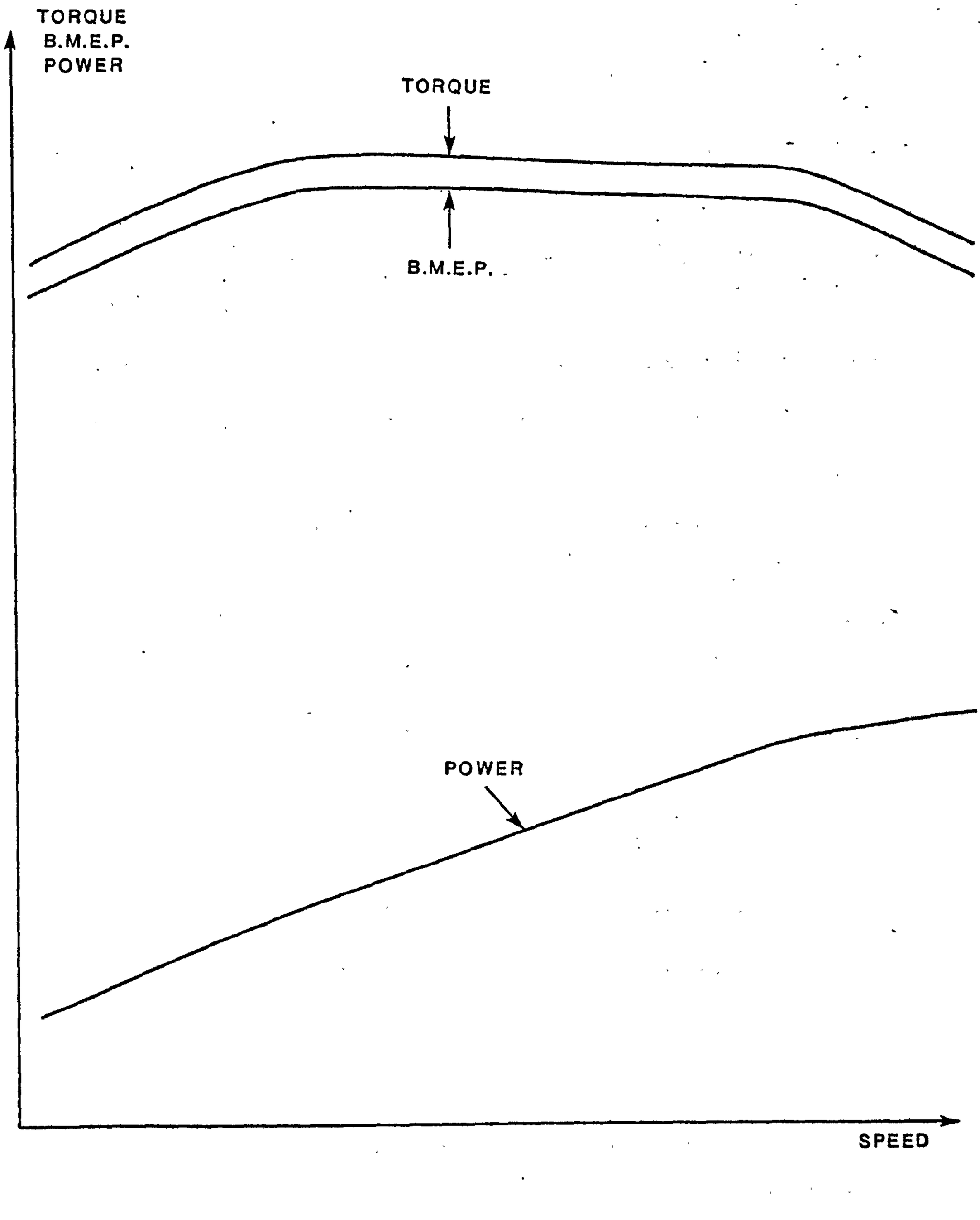


FIGURE 7.3 - FULL LOAD SIMULATION RESULTS

the set point values are used also as the process values), this simulation does not include any form of control. This was done deliberately as one of the major uses of the system was to investigate the use of direct digital control of a test bed.

The inputs to the simulation are throttle position and dynamometer load current (or dynamometer controller output). The basic dynamics of the system assumes that the engine and dynamometer speeds are equal (i.e. the propellor shaft twist is constant):-

$$\text{Thus} \qquad \dot{\theta}_E = \dot{\theta}_D = \dot{\theta}_{ED} \qquad (7.5)$$

$$\text{and} \qquad Q_E - Q_D = I_{ED}\ \ddot{\theta}_{ED} \qquad (7.6)$$

where Q_E = Torque developed by engine

Q_D = Torque developed by dynamometer

I_{ED} = Engine/dynamometer system inertia

$\ddot{\theta}_{ED}$ = Engine/dynamometer system acceleration

As stated earlier, the speed/torque characteristic of an internal combustion engine can normally be simulated by a quadratic equation:-

$$Q_E = K_1 + K_2\,\dot{\theta}_{ED} + K_3(\dot{\theta}_{ED})^2 \qquad (7.7)$$

In the case of a diesel engine, K_1, K_2 and K_3 are constants

and are not dependent on the throttle position, ϕ , as is the case for petrol engines. With diesel engines, however, it is necessary to take into account the operation of the governor. After the governor cut-off speed, $\dot{\theta}_{EC}$, has been exceeded the fuel to the engine is progressively cut-off so that the developed torque declines rapidly. This behaviour can be described by the following equation:-

$$Q_E = Q_{EC} - B(\dot{\theta}_{ED} - \dot{\theta}_{EC})^2 \qquad (7.8)$$

$$\text{for} \quad \dot{\theta}_{ED} \geq \dot{\theta}_{EC}$$

where Q_{EC} is the engine developed torque at the cut-off speed, i.e. :-

$$Q_{EC} = K_1 + K_2 \dot{\theta}_{EC} + K_3(\dot{\theta}_{EC})^2 \qquad (7.9)$$

Although the basic quadratic equation 7.7 is not effected by throttle position, ϕ , $\dot{\theta}_{EC}$ and B are both complex functions of ϕ. In the simulation they are generated by using two variable diode function generators (V.D.F.G). Their functions are shown in Figures 7.4 and 7.5. Examining the complete simulation shown in Figure 7.6, it can be seen that for $\dot{\theta}_{ED} < \dot{\theta}_{EC}$, the analog switch disconnects the second term of equation 7.8 and the track/store amplifier (no. 26) operates according to equation 7.7. When the cut-off point is exceeded the track/store amplifier will enter the store mode thus preserving the value of Q_{EC} whilst the analog switch will be on

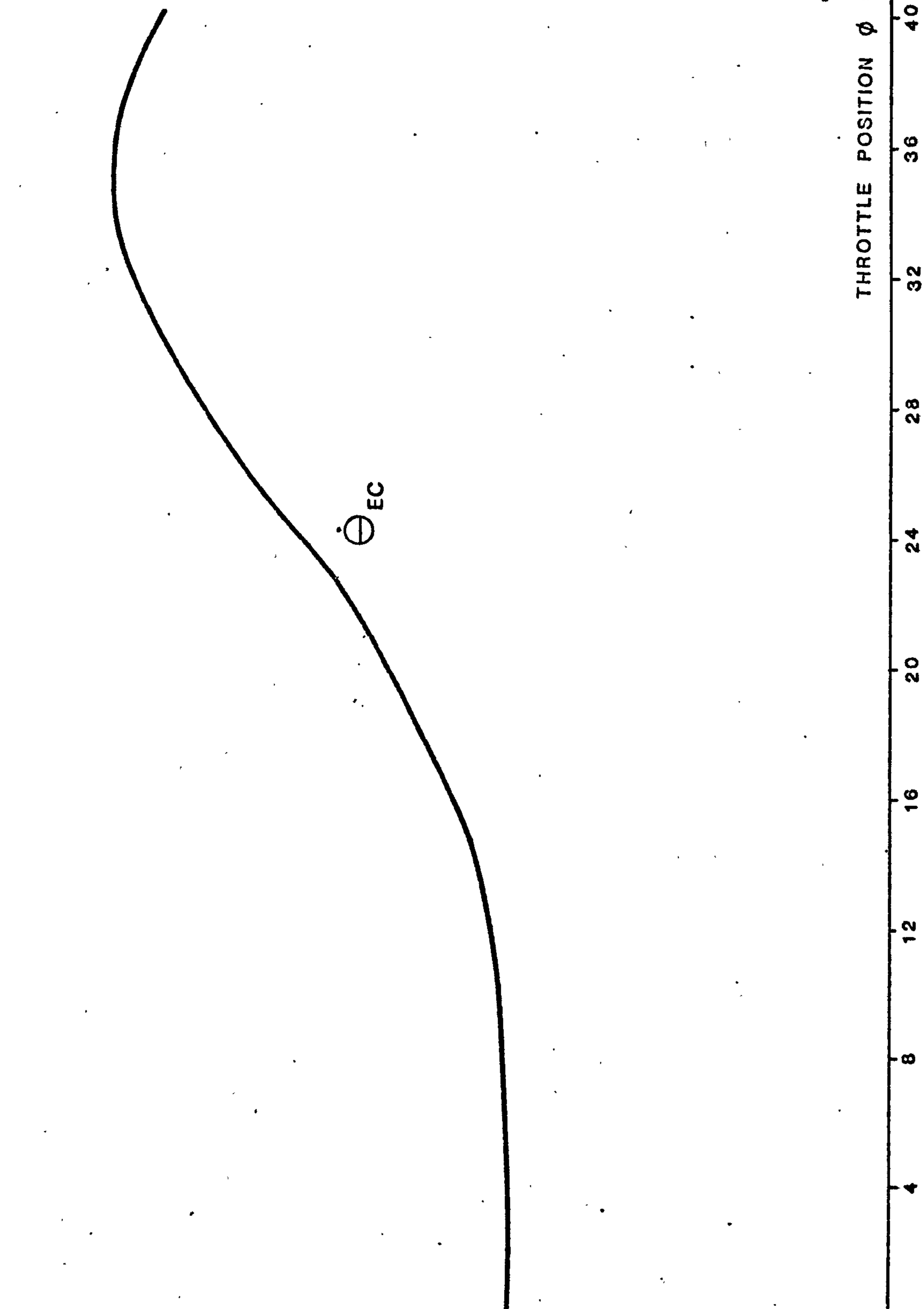


FIGURE 7.4 - GOVERNOR CUT-IN SPEED AGAINST THROTTLE POSITION

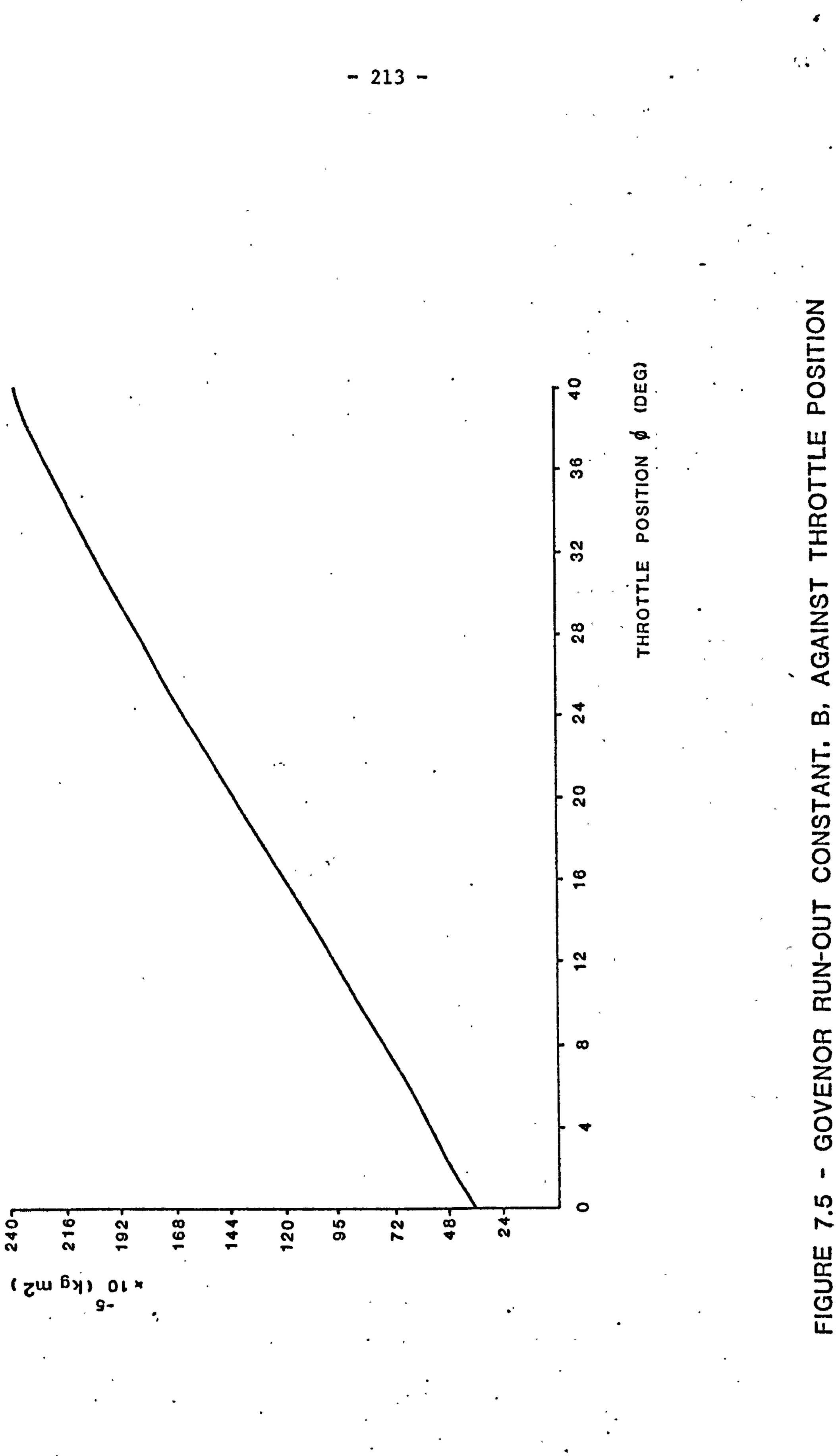


FIGURE 7.5 - GOVENOR RUN-OUT CONSTANT. B, AGAINST THROTTLE POSITION

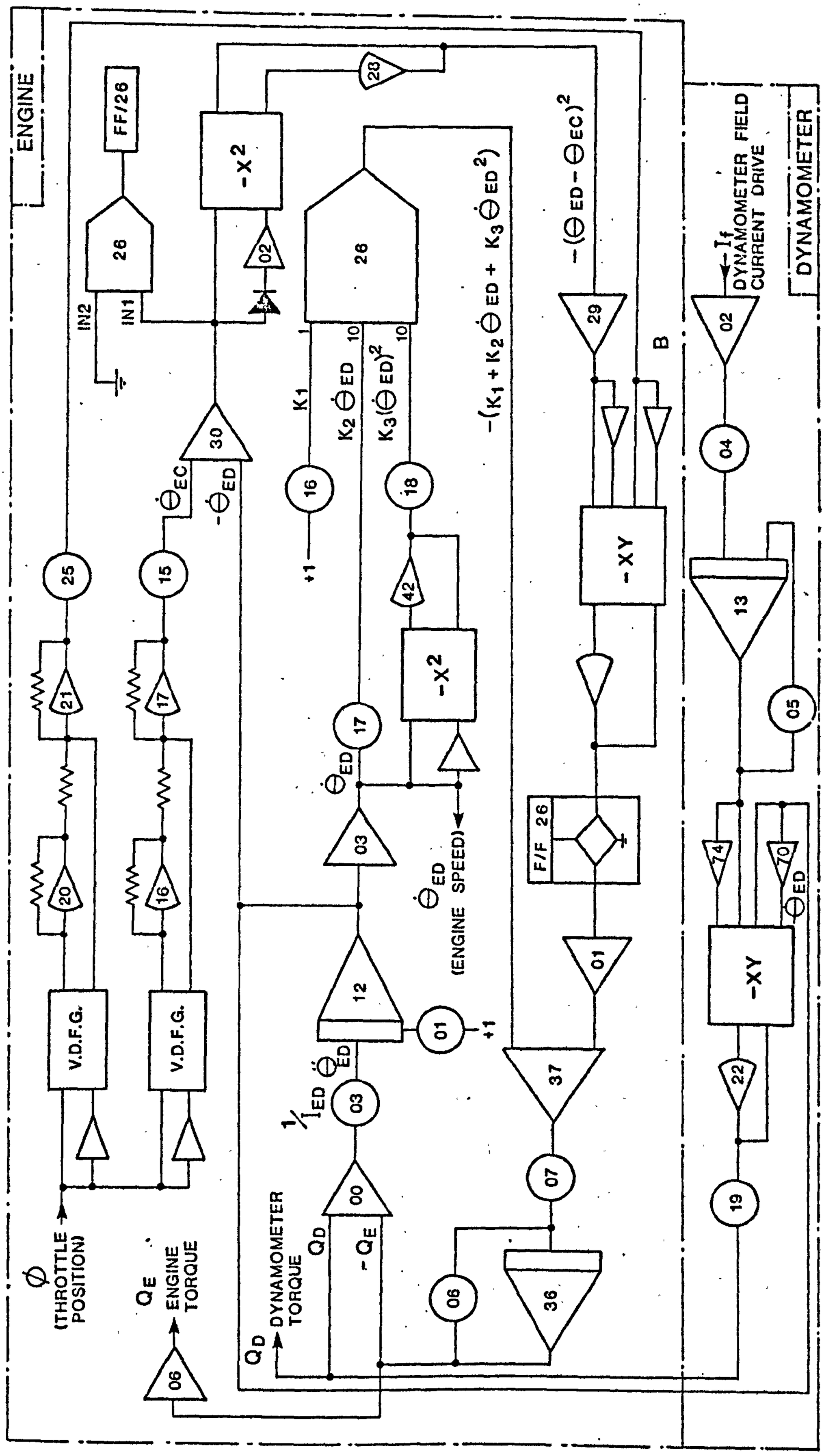


FIGURE 7.6 - ENGINE / DYNAMOMETER SIMULATION

to allow the complete simulation of equation 7.8. The developed torque is subjected to a first order lag before being used, together with Q_D, to solve equation 7.6 for the engine speed, $\dot{\theta}_{ED}$. The speed/torque behaviour of the simulation is shown in Figure 7.7.

A basic dynamometer loading function is described fairly simply by the relationship:-

$$Q_D = K_D (I_f \cdot \dot{\theta}_{ED})$$

where I_f = Dynamometer Field Current.

In addition the dynamometer is represented as being subject to a first order lag.

This simulation provided the mechanism for digital speed control loop exercises. The throttle position simulation is a steady state one and does not allow for lags in the system.

7.4 Digital Simulation

This section of digital simulation includes two different sections covering the different halves of a single complete simulation package. One half concerns the simulation of various processes whilst the other is about a program to perform direct digital control (DDC) by simulating an analog three term controller. Both parts could be used independently and indeed the DDC program was used in connection with the analog engine and dynamometer simulation described in the previous section.

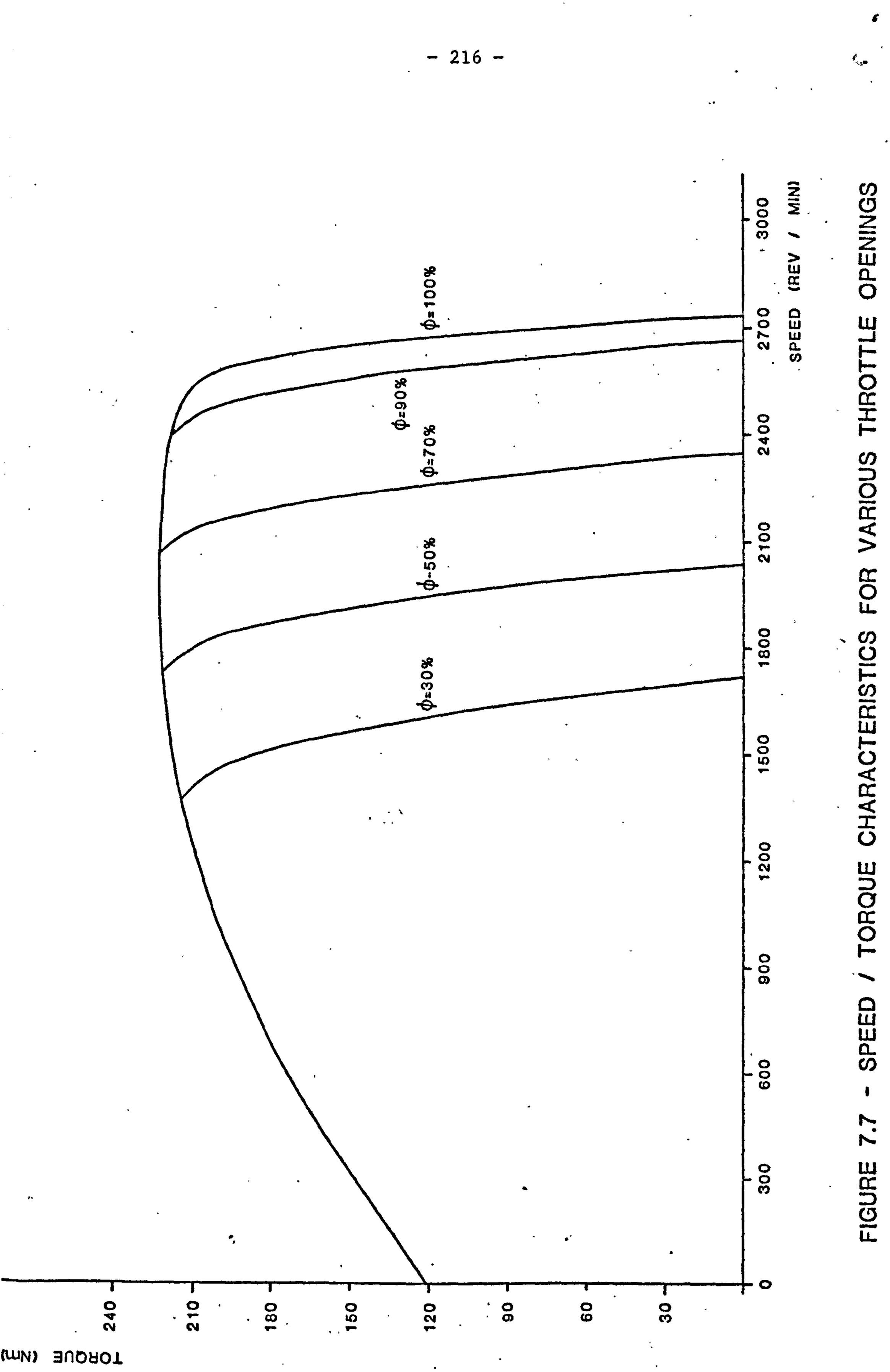


FIGURE 7.7 - SPEED / TORQUE CHARACTERISTICS FOR VARIOUS THROTTLE OPENINGS

7.4.1 - Digital Process Simulation

For the most part it is not difficult to simulate digitally any process that can be described in mathematical terms. In the case of most of the equations in the earlier part of the chapter, it would simply be a matter of writing the equivalent statements in some language such as Fortran. Problems arise however when the question of parallel operation occurs. If one of the inputs to a section of a process is a function of its current output, then how can the output be calculated if the input is unknown? The answer is often the use of an approximation and iterative solution. This is illustrated by the example below.

Many real systems and processes behave as second order systems and it was decided to write a computer program to simulate a general second order system. The system can be described by the equation:-

$$\frac{Id^2\theta_o}{dt^2} + C\frac{d\theta_o}{dt} + k\theta_o = \theta_i \qquad (7.11)$$

where θ_i and θ_o are the input and output of the system

I = Inertia

C = Damping factor

k = Spring Stiffness

Rewriting the equation and using the D operator gives us:-

$$D^2\theta_o + R_2D\theta_o + R_3\theta_o = R_1\theta_i \qquad (7.12)$$

where $R_1 = 1/I$

$R_2 = C/I = 2 \xi W_n$

and $R_3 = k/I = W_n^2$

with W_n being the system natural frequency and ξ the damping ratio.

The analog type solution to this equation is shown in Figure 7.8. The overall function is given by:-

$$D\theta_o = \int (R_1\theta_i - R_2 D\theta_o - R_3\theta_o)\, dt \qquad (7.13)$$

and

$$\theta_o = \int (D\theta_o)dt \qquad (7.14)$$

It can be seen that in this case we have the problem that the term θ_o, that we wish to solve for, cannot be separated. The algorithm of the process simulation program is shown in Figure 7.9. The variables in the algorithm are defined as follows:-

OUT = θ_o

IN = θ_i

F1 = $R_1\theta_i - R_2 D\theta_o - R_3\theta_o$

RINT1 = $D\theta_o$

RINT2 = $\int D\theta_o\, dt$

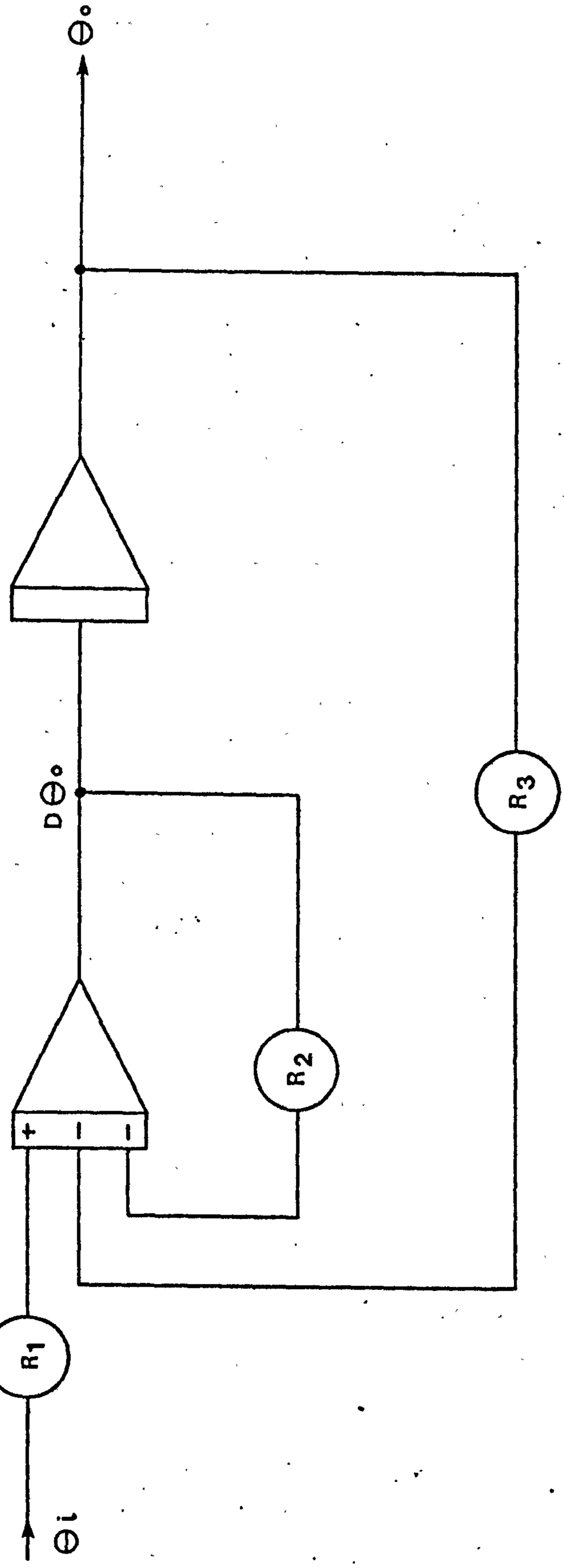


FIGURE 7.8 - SECOND ORDER SIMULATION

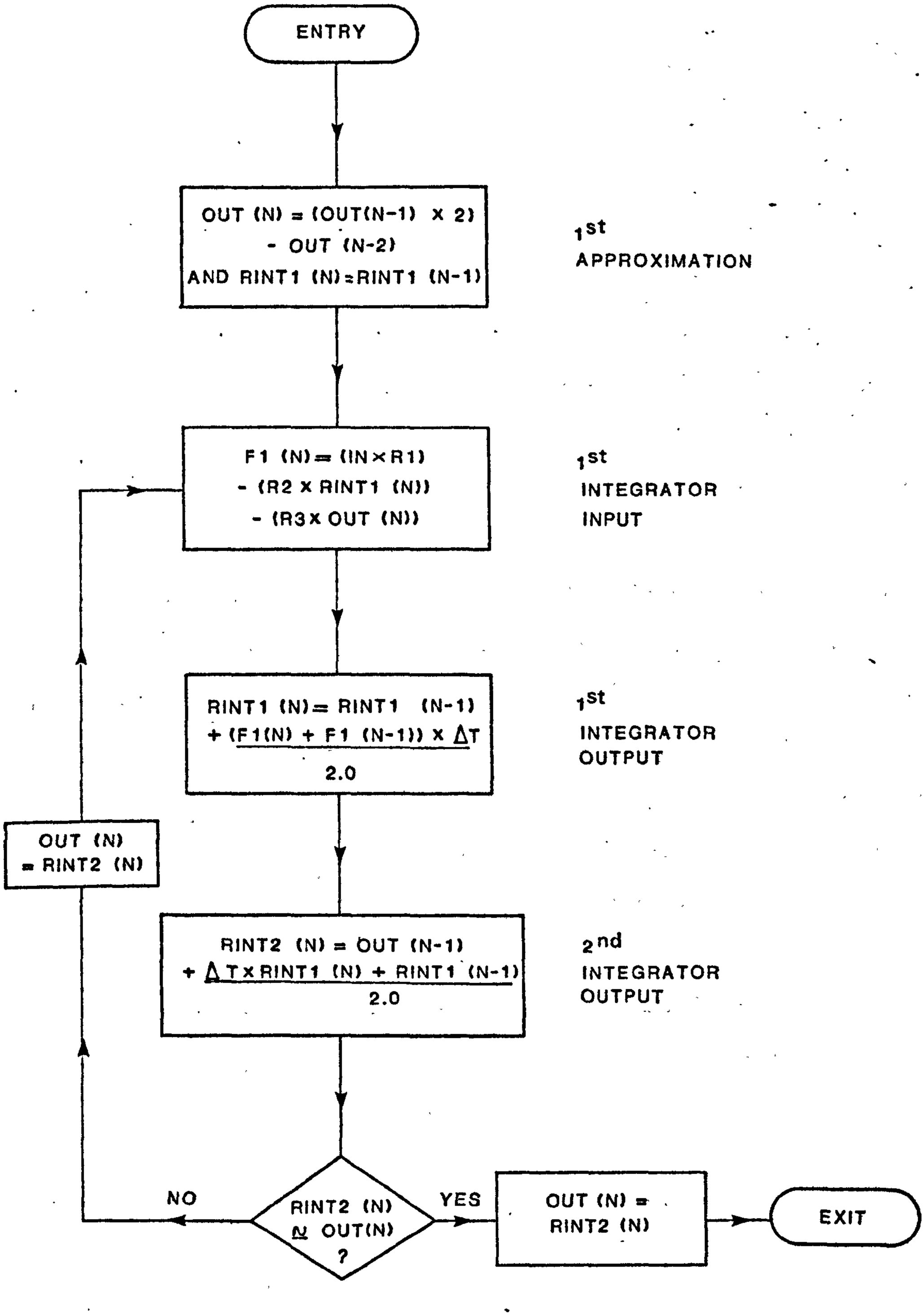


FIGURE 7.9 - PROCESS SIMULATION ALGORITHM

and the parameter X(N) is that parameter current value whilst X(N - 1) was the value at a time ΔT before.

The program calculates the process output each 10 millisecond (simulated) interval. The first stage is to assume that the value of $D\theta_o$ is equal to its previous value and to calculate an approximation for the current value of θ_o using linear interpolation based on its two previous values. These approximate values are used to calculate the current sum of the inputs to the first integral term. It is then assumed that the input to the integrator varies linearly between successive process output calculations, allowing the current output, which is the new value of $D\theta_o$, to be calculated. This is again assumed to behave linearly over the interval and is used to perform a second simulated integration. The result of this gives a new approximation for θ_o that can be used to repeat the above calculations until the difference between successive approximations becomes less than the required accuracy of the simulation.

The program has the form of a subroutine and calculates the change in the process output for a 10 millisecond interval. Its use in a totally digital simulation is illustrated later on. The technique replaces the standard methods of solution not available on the minicomputer used for the work.

7.4.2 - Digital Control

The implementation of a three term (proportional, integral and derivative) controller in a digital processor can also to some extent be regarded as a simulation exercise as it is basically representing analog circuits using artificial means.

Unlike its analog counterpart, the digital computer has no direct method of calculating the instantaneous value of the integral and the derivative with respect to time of a varying function. Thus it is necessary to use some form of approximation. The overall function of a three term controller can be represented by equation 7.15:-

$$V = K_1 \theta + K_2 \int \theta \, dt + K_3 \frac{d\theta}{dt} \qquad (7.15)$$

where V = Controller Output

θ = Error = $\theta_i - \theta_o$

and K_1, K_2 and K_3 are the proportional, integral and derivative gains.

If the controller program is run at regular intervals of duration Δt, the three term operation can be simulated as follows; at the end of each interval the current process and set point values are used to calculate the current error θ_n. The value of the integral term is found by assuming the error changed linearly over the last time interval (see Figure 7.10) and the value for the integral term calculated by the same method as in the previous section. Thus we are assuming the following equation:-

$$\int_{t=0}^{t=t_n} \theta \, dt \simeq \int_{t=0}^{t=t_{n-1}} \theta \, dt + \left[\frac{(\theta_n + \theta_{n-1})}{2} \times (t_n - t_{n-1}) \right]$$

(7.16)

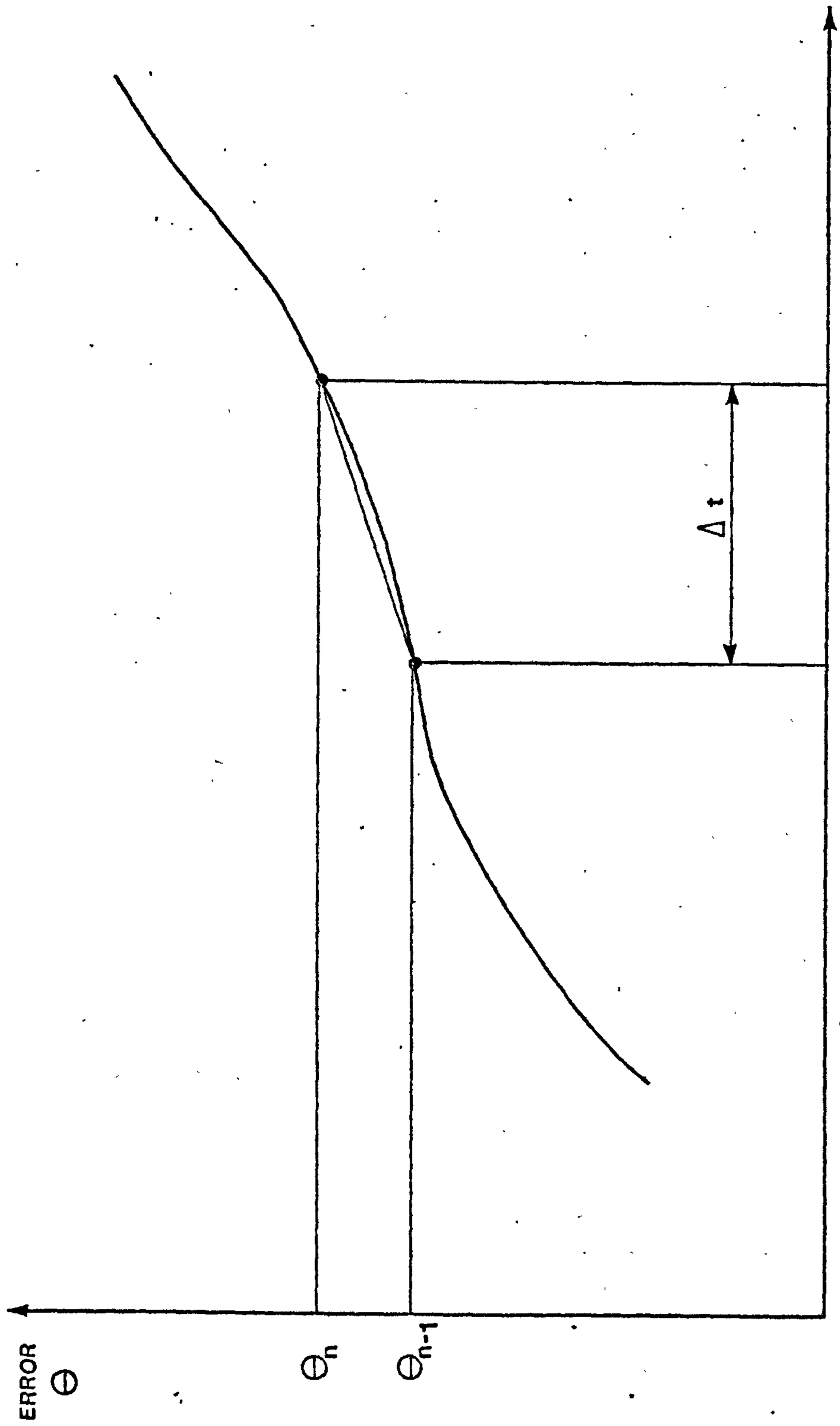


FIGURE 7.10 - ERROR APPROXIMATION

Similarly the differential term is assumed to be the average slope over the last time period, so that:-

$$\frac{d\theta}{dt} \simeq \frac{\theta_n - \theta_{n-1}}{t_n - t_{n-1}} \qquad (7.17)$$

Given these approximations, it is then a simple matter to use them to calculate the controller output according to equation 7.15 using pre-defined values of K_1, K_2 and K_3.

7.4.3 - Digital Simulation Example

The use of digital simulation techniques can be illustrated by an example based on the two simulations previously mentioned.

The purpose of the simulation was to examine the behaviour of the direct digital control algorithm when used in conjunction with a throttle actuator. The actuator was represented by a second order system using the digital simulation. The overall package algorithm is shown in Figure 7.11. First an array of set points was generated to cover the duration of the simulation which was 5 seconds. The user could generate step, ramp and sine-wave functions in the set point. The next stage was to enter the values needed by the simulation to define the throttle actuator. The values used were those of the actuator used on one of the test rigs at the College, whose natural frequency and damping ratio were known. The controller gains and the frequency of execution of the control loop had to be specified. After setting up specified initial conditions which assume that the

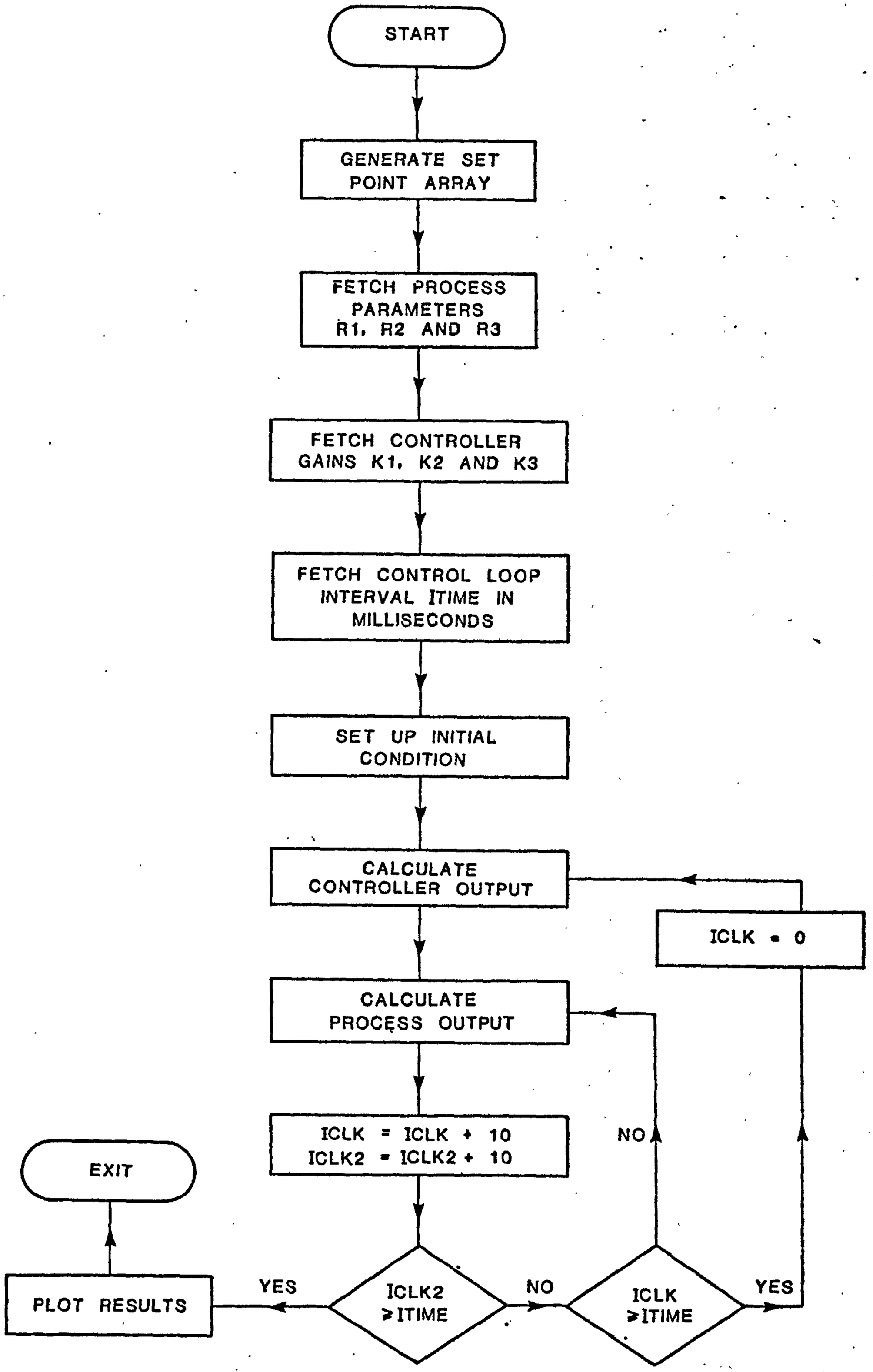


FIGURE 7.11 - DIGITAL PROCESS AND CONTROL SIMULATION

system has stabilized with zero error some time previously, the simulation calculation begins. The value of the process output is calculated for 10 millisecond intervals over a period of 5 seconds and, at the relevant points depending on the set interval, the controller output is re-calculated. When this has been done (because of the amount of processing required it takes considerably more than 5 seconds of computer time), the program plots the various curves showing how the simulated system behaved. This output is shown in Figures 7.12 and 7.13 where the set-point and process output values are plotted against time for two different values of derivative gain. It is also possible to have the computer plot the controller output values.

7.5 Summary

One of the most frequent objections against the use of simulations in developing automated test systems is that the work developing the simulation in itself is excessive. This chapter is intended to illustrate the way in which a simplified simulation, which is not difficult to design, can be used advantageously. It is not claimed that such a simulation can enable the development of the test system all the way, but it can be useful in the earlier stages before transferring to using the actual process itself for the final stages. For a majority of cases it is, at present, best to use analog simulation techniques with their ability to operate in real time. On the other hand in cases of background or theoretical inyestigations where actual processing time requirements are not vital, the use of a digital computer to proyide the simulation environment has its advantages.

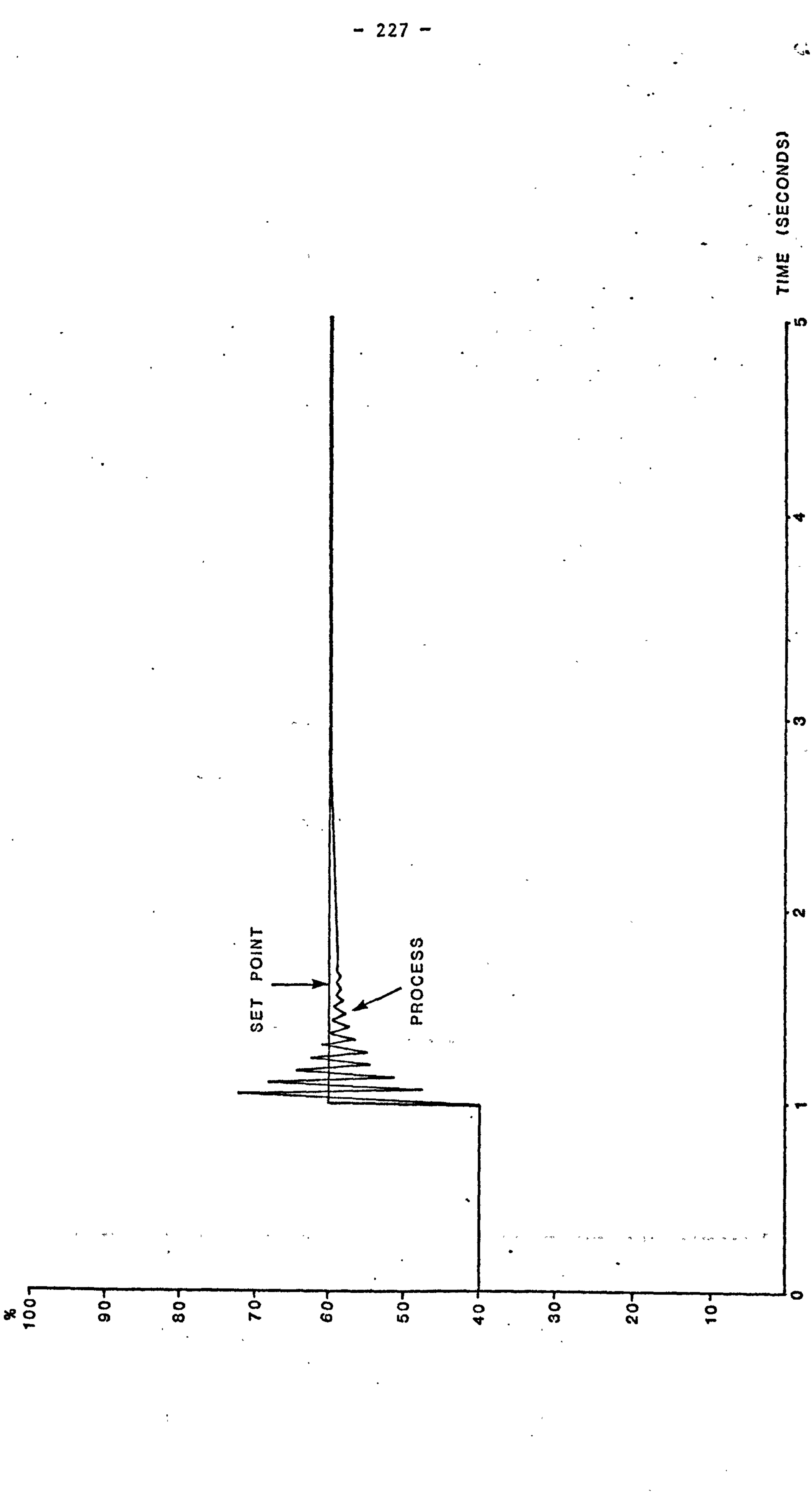
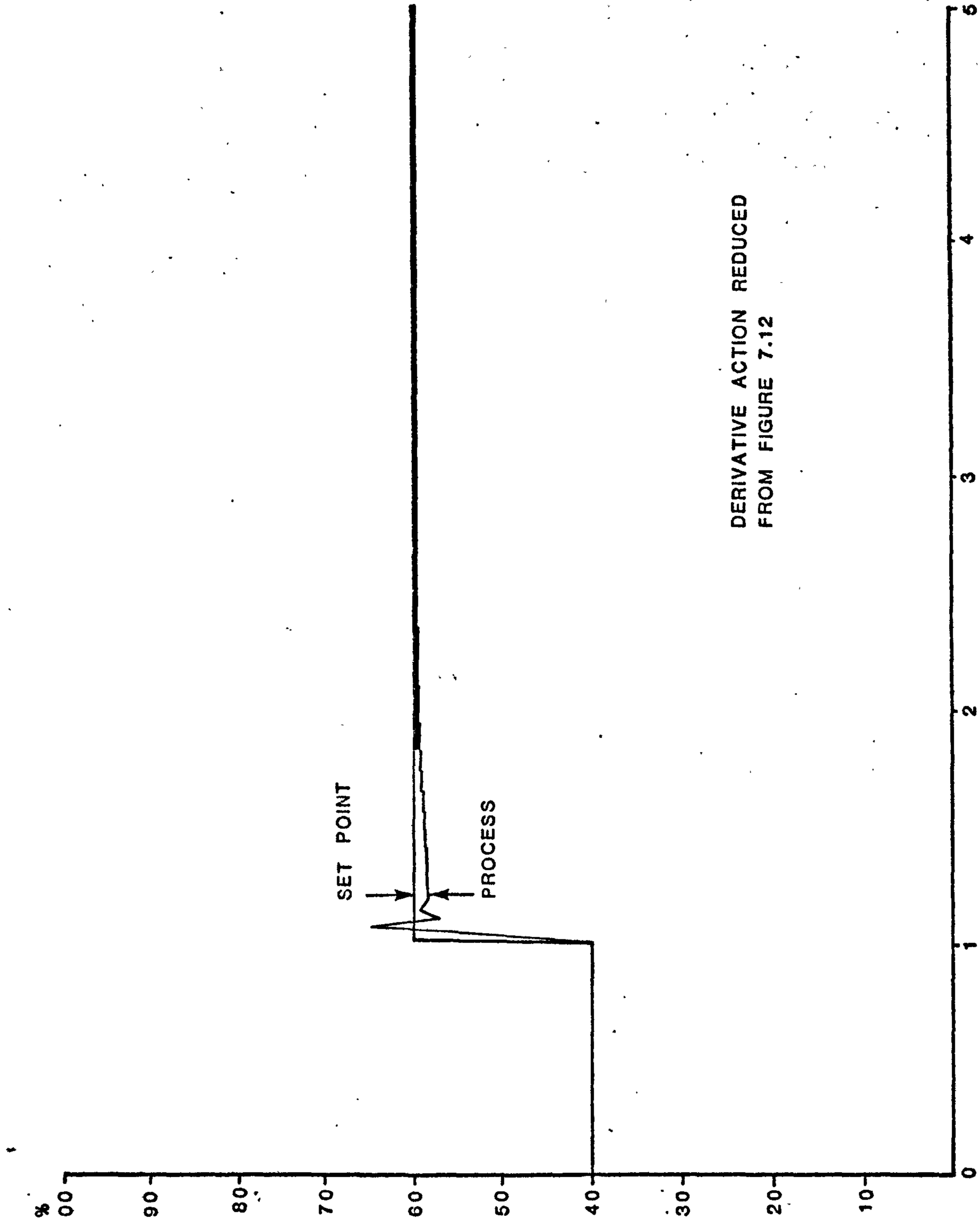


FIGURE 7.12 - SET POINT AND PROCESS VALUE AGAINST TIME


TIME (SECONDS)
0
1
2
3
4
5
TIME
FIGURE 7.13 - SET POINT AND PROCESS VALUE AGAINST TIME
DERIVATIVE ACTION REDUCED
FROM FIGURE 7.12
SET POINT
PROCESS
%
100
90
80
70
60
50
40
30
20
10
0

The chapter describes two analog simulations, one fairly complex and the other much simplified. It also shows how continuous analog processes may be simulated by digital methods and demonstrates this for a second order system and for three term control.

Chapter 8

INTERACTIVE HANDLING SYSTEM SIMULATION

8.1 Introduction

In the previous chapter we considered the use of simulation techniques to assist in the development of process control systems. However there is another way in which simulation can be of vital importance in the field of automation of engine testing. In a production test environment, we are usually basing our transport of engines from production areas to test benches and on to rectification or finishing, on some form of automatic handling system. These handling systems come in many different types. Perhaps the two most common systems traditionally are the overhead chain conveyor and the ground-based dragline. More recently we have seen the advent of more advanced concepts such as the track based controlled pallet (58) and the automatic independent trolley (41).

Whatever the form of handling system used, it is very important that its design should be optimised to allow maximum throughput. Failure to do this will result in the requirement for more test beds and hence higher capital and running costs. To plan this optimum layout we will need some form of simulation to examine the behaviour of different handling system layouts. It should be noted that this function cannot be divorced from the actual test bed design work, as the type of test performed will effect its duration and hence the overall performance of the facility. Although little is said here about the actual tests, the handling system layout design is very much dependent on them.

8.2 Conventional and Interactive Simulations

The first thing to become obvious about any simulation of material handling systems is that we are using it to study a system that may not even exist as yet and hence there is no constraint to running the simulation in real-time. For this reason the use of a digital computer based solution is ideal.

When the problem of using a simulation to provide an optimising tool for production test facility design was first approached, the possibility of using one of the many standard simulation packages was condsidered. However after some investigation it was decided that it would be better to design a simulation specifically for the task of providing interactive simulation of material handling systems. The reasons for rejecting the use of a standard system were twofold. Firstly the level of expertise needed to generate a given simulation is considerable and tends to be based around computer programming type techniques (33, 42). The objective of the new simulation would be to provide a configuration method based on data tables rather than programming.

The second reason lies in the method of operation of the simulation. In the conventional manner the simulation tends to be based on statistical analysis rather than the actual simulation of the physical system. The system tends to be operated by the submission of a package of programs together with some data and after execution the resulting processed data is returned to the user. This system provides little or no opportunity for the would-be designer to interact dynamically with his simulation. It was felt that a

more useful approach would be to provide a simulation that actually traced the behaviour of a real system by running through its operation with a time scale that was a speed-up version of real time. The statistical data gathering would be provided as a background operation and the user could communicate with the simulation at any time to provide the ability to interrogate and/or alter the simulation. Thus the concept of an interactive simulation allows the user to observe the validity of the simulation and to use his own engineering skills to optimise his handling system design.

8.3 Simulation Package

The package of programs that are used to implement the simulation are by and large written in FORTRAN. The package was originally implemented and run on an EAL 640 computer. This is a 16-bit machine and the program could also be run on any other machine supporting a FORTRAN compiler and the necessary input/output capabilities. The computer used has a memory size of 16K words and although it was possible to run the package within this amount of memory, this did provide a severe constraint on the amount of data storage available and hence the complexity of handling system that could be simulated. This limitation is not inherent in the software and given the availability of a larger memory, very complex systems could be simulated.

The operation of the simulation is based on a series of data tables. The most important of these is the track layout table.

8.3.1 Track Simulation

The track simulation data table forms a numerical description of the physical layout of track. This table can normally be regarded as static data in that it is not altered by the computer although it can be accessed by the user by way of the interactive features of the package.

To enable any form of layout to be described, the simulation defines six different basic track types. The way in which these can be put together to implement the complete layout will be shown later.

The different types of track are shown in Figure 8.1. Each section of track in the simulation has a total of seven items of data associated with it as shown in Figure 8.2.

i) Type 1 - Ordinary track.

This is the normal section of track down which material units (in our case, engines) move. The data items for this type are described below:-

a) Following Track Section - this is the number of the next track section that an engine enters on leaving the current section.

b) Track Type Number - the tag number identifying what sort of track the entry is for.

c) Length - In terms of the simulation the length of a given section of ordinary track has two different meanings. Given a value of length X, this will mean that an engine moving freely along the track will take X time periods to traverse this section. It also

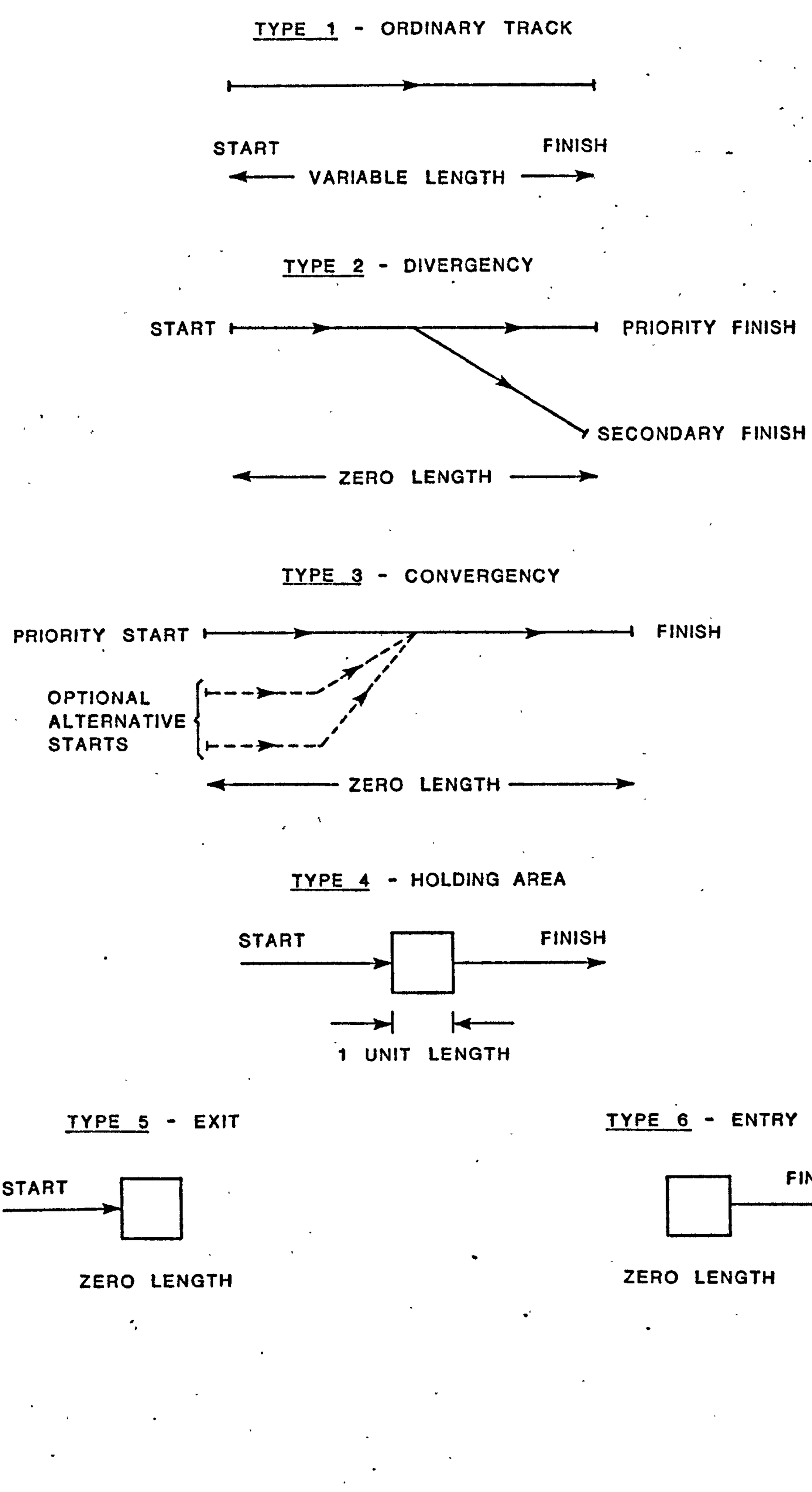


FIGURE 8.1 - TRACK TYPES

DATA ITEM / TRACK TYPE	1	2		3	4	5	6	7
ORDINARY TRACK	FOLLOWING TRACK SECTION	TRACK TYPE NUMBER	1	LENGTH	SPECIAL ACTION FLAG	SECTION BLOCKED FLAG	X-AXIS COORDINATE OF START	Y-AXIS COORDINATE OF START
DIVERGENCY	PRIORITY FOLLOWING TRACK SECTION		2	DIVERSION STATUS	SECONDARY FOLLOWING TRACK SECTION		NO FUNCTION	NO FUNCTION
CONVERGENCY	FOLLOWING TRACK SECTION		3	NO FUNCTION	PRIORITY PREVIOUS TRACK			
HOLDING AREA			4	HOLDING TIME			X-AXIS COORDINATE OF LOCATION	Y-AXIS COORDINATE OF LOCATION
EXIT	NO FUNCTION		5	REFERENCE NUMBER	SPECIAL ACTION FLAG			
ENTRY	FOLLOWING TRACK SECTION		6	NO FUNCTION		ENTRY STATUS		

FIGURE 8.2 - TRACK DATA FIELD

means that should the section begin to fill up with a queue, due to a hold-up somewhere ahead, then X engines will be the total queuing-capacity of the section. Obviously at first sight this is unsatisfactory as the two parameters are not normally the same for a section of track. However in reality only one of the two criteria will be applicable to describing the behaviour of a given section of track. In one case we have a section of track that is rarely loaded to anything like capacity and hence as long as the normal free transit time is accurately reflected by the value given for the length, then the simulation will remain valid. On the other hand some sections of track will have a high density usage with a low flow rate (such as engine storage buffers, etc.). Here the rate at which an engine moves through the section will tend to be controlled by the rate at which engines are released from the finish, at which point all the remaining engines are shunted one place forward. Setting the length equal to the actual number of engines the section can accomodate will maintain the accuracy of the simulation.

In cases where there is a special need to simulate both parameters simultaneously there are more complex methods available that make this possible (see the repair area section of the simulation described later on in this chapter).

d) Special Action Flag - this is the mechanism by which various non-adjacent sections of track can interact with each other. For instance, the example later in the chapter shows how an engine leaving a test cell can alter the flow at some point distant to itself to divert an untested engine towards itself. The special action

flag takes the form of an integer constant. If the value of the flag is zero then no special action is associated with that section of track. If the flag has any other value then a call is made to the special action control subroutine (SPECL). If the number is negative then this call is made whenever an engine leaves that section of track. If the number is positive then the call is made whenever an engine enters the track section. The control subroutine decodes the actual flag number together with information on the track section and engine concerned and makes a call to the required routine in a group of special action subroutines. These perform functions such as blocking and unblocking track sections (see below) and also updating various background statistical monitoring facilities.

e) Section Blocked Flag - if this parameter has any value other than zero then it will not be possible for any engine to enter the section of track regardless of whether there is room for it or not. Setting and resetting the flag by way of special action routines can provide remote control of switching points. Sections can also be blocked or un-blocked at any time by the user through the interactive command repertoire in order to allow the effects of breakdowns and other failures to be examined.

f) and g) Graphical Coordinates - a pair of numbers used to locate the position of a track section when using the graphic plotting capabilities of the system's visual display unit (V.D.U.). These values are for display purposes only and do not need to reflect the real location of a track section.

The above data parameters are sufficient to define fully a section of ordinary track. Other types of track are detailed below but where the data item is identical in function to the above items it is not listed. Some types of track have less than the seven items and dummy items with no function are also not listed.

ii) Type 2 - Divergency

Where a track path diverges into two separate paths then the three sections (one sending and two receiving) are linked by a section of this type. The section is defined as having zero length, i.e. an engine will apparently pass directly from the preceeding to the subsequent section in one move. However this dummy track section must be included in the list of parts of the layout as its functions control the behaviour of engines passing through the switching point.

a) Priority Following Track Section - this is the number of the track section that an engine will be transferred to unless diverted (see below).

c) Diversion Status - each engine has a status number associated with it, normally used to represent its condition (e.g. untested, tested-failed or tested-passed). Normally an engine will be passed to the priority following track section, but if its status number is identical to the diversion status given for the particular section of track an attempt will be made to divert it instead to the secondary following track section. If this attempt fails, i.e. the alternative section is already full of engines or has its section blocked flag set, then the engine will instead proceed to the original priority

following track section.

It should be emphasised that there is no reason why the diversion status, as well as any other parameter, cannot be altered dynamically either as a result of interactive user operation or calls to special action routines. This could for example be used to switch two different types of engine alternatively onto a spur.

d) Secondary Following Track Section - is the number of the track section that is the target should the engine and track diversion status match.

iii) Type 3 - Convergency

If there is no requirement for a priority source, then two or more tracks can be converged simply by giving the number of the single subsequent section as their Following Track Section. In this case should two engines both arrive at the convergency within the same time period the order in which they will proceed is arbitrary (it in fact depends on the relative position of the two engine's data in the storage buffer). To provide priority it is possible to use another of the dummy track sections of zero length. Any number of alternative sources can be used as the starting point of the convergency and engines arriving there are transferred straight to the next section track after the end of the convergency. However any engine arriving by the priority track will be sent first, with other engines arriving at other start points being held until the next time cycle.

In general it is often possible to deliberately misuse track types in order to simulate functions not directly catered for. An

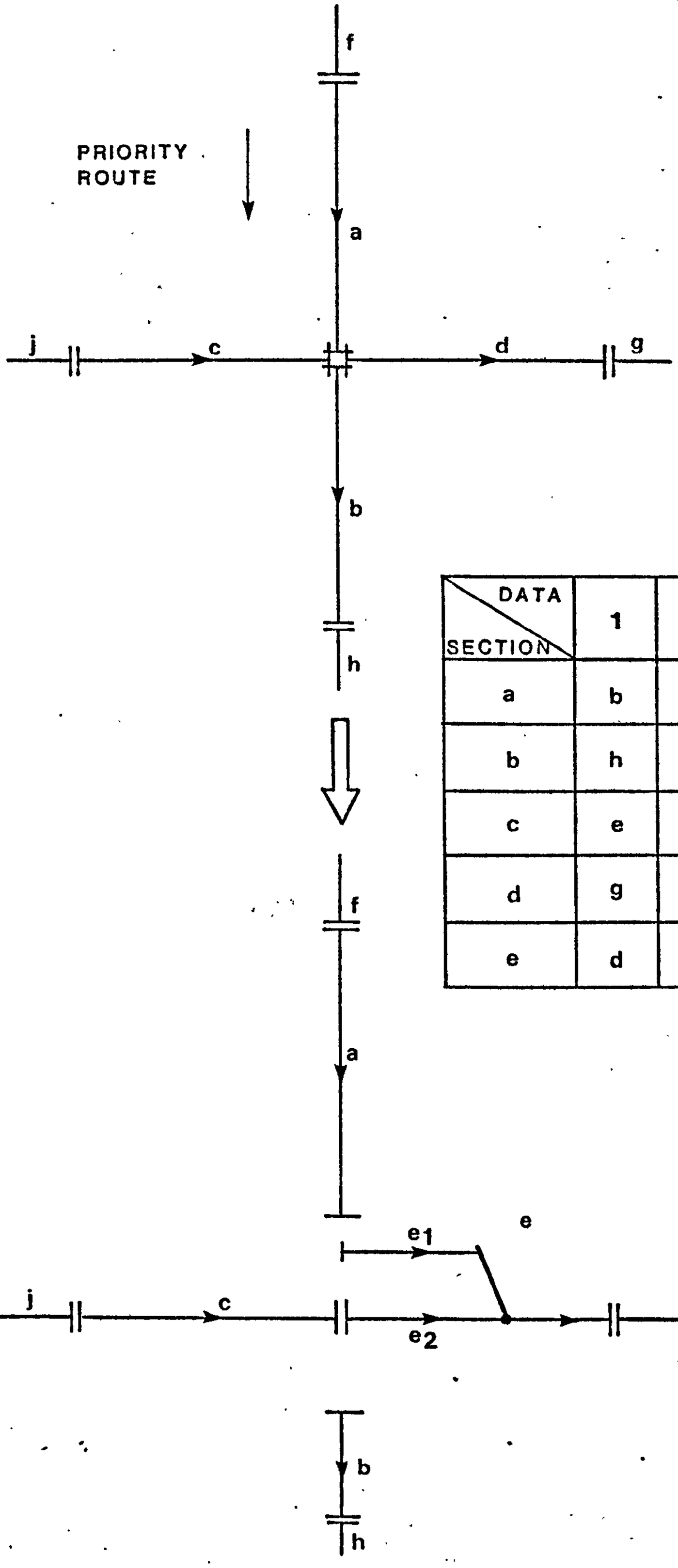
example of this is shown in Figure 8.3 where the simple cross-over point of two tracks where there is a right of way rule can be simulated with the aid of a convergency. Although engines pass directly from section a to section b and also from c to d, those going from a to b have priority. The use of the dummy convergency, section e, simulates this. Engines leaving section a will always pass to b because the actual simulation control is forward looking. However when an engine is ready to leave section c, the non-existent priority route a → e_1 → d will stop the transfer if there is an engine about to leave section a.

d) Priority Previous Track - is the number defining which of the track sections sending engines to the convergency has priority.

iv) Type 4 - Holding Area

The type of track section known as the holding area provides the facility to hold an engine in one place for a given amount of time. The holding area could be regarded as an ordinary track section differing only in that instead of its transit time and storage size being identical, the transit time is defined by the holding time, whilst the physical size (number of engines that can be contained) is always equal to one. The commonest usage in our application is the simulation of test cells.

c) Holding Time - the number of time periods for which the engine is to remain in the holding area.



SECTION \ DATA	1	2	3	4
a	b	1	–	–
b	h	1	–	–
c	e	1	–	–
d	g	1	–	–
e	d	3	–	a

FIGURE 8.3 - CROSS OVER USING A CONVERGENCY

v) Type 5 - Exit

All the types of track section described so far do not create or terminate the existence of an engine, they merely take it from one track section and ultimately pass it on to another. However any real system must have a start and finish or indeed, perhaps several of each. When an engine reaches a track section of type 5 it will be deleted from the list of engines in the system. Unlike transfer through holding areas representing test cells, etc. where special action flags must be used, the transfer of an engine to an exit automatically results in the activation of certain statistical analysis and reporting facilities as described later on.

c) Reference Number - this is an arbitrary but unique reference number for each exit point to allow its identification by the statistical analysis facilities.

vi) Type 6 - Entry

As a counterpart to the exit we must also have some point at which new engines can enter the sytem. As can be seen later on entry points are not processed as part of the general system update algorithm. Instead at the beginning of every update cycle, each of the entry points defined in the system will attempt to create a new engine in the first space (temporal and physical) of the track section defined by its own data item 1. Individual points will be disabled if their following section is blocked either by an engine or by a section blocked flag.

If the above conditions are both false then a check will be made to see if there is room in the engine data storage buffer to generate a new engine. The limit to the number of engines allowed to exist at any one time can be set by the user. Only if all the above requirements are met will a new engine be created.

e) Entry Status - as explained earlier, each engine has a status number assigned to it which controls its behaviour at divergencies. The number defined as the entry status is the initial status given to each engine created at that entry point.

8.3.2 - Engine Simulation

The previous section shows how the track data provides the static background against which the simulation acts. In the foreground we have the dynamic data that describes the engines and alters as they move through the system. Each engine has a total of five items of data associated with it. The information is stored in the engine data storage buffer. The different data items in each entry in the buffer are:-

i) Reference Number - is a unique tag given to each engine created by an entry point. The first engine created after the start-up of the simulation is number 1, the second number 2 and so on. A vacant area in the data storage buffer is identified by having the first data item equal to zero.

ii) Present Section - the number of the track section that the engine is currently in.

iii) Time/Length to go - represents the temporal, and in the case of ordinary track, physical position of the engine within its current track section. When an engine first enters either a type 1 or type 4 track section, this item is set equal to the value, contained in data item 3 (see Figure 8.2) of the tracks data entry. This item is then used as a down-counter until zero is reached and the engine can proceed to the next track section.

iv) Present Status - a number signifying the present status of the engine, i.e. untested, tested, etc. The actual number used to represent any given condition is irrelevant as long as the corresponding diversion status numbers in the track simulation are made to match.

v) Entry Time - the time at which the engine entered the system (was created). This is used by part of the statistical monitoring facilities.

8.3.3 - Simulation Algorithm

Having examined the way in which the elements of the simulation are reduced to the data tables that describe them, we can turn to examining the routines that use them to operate the simulation.

The overall algorithm of the simulation is shown in Figure 8.4. On commencement, the first action is to set up the data tables necessary to define the particular layout to be simulated. This information is fed in from a paper tape that can be prepared off-line. The parameters on the tape are as follows:-

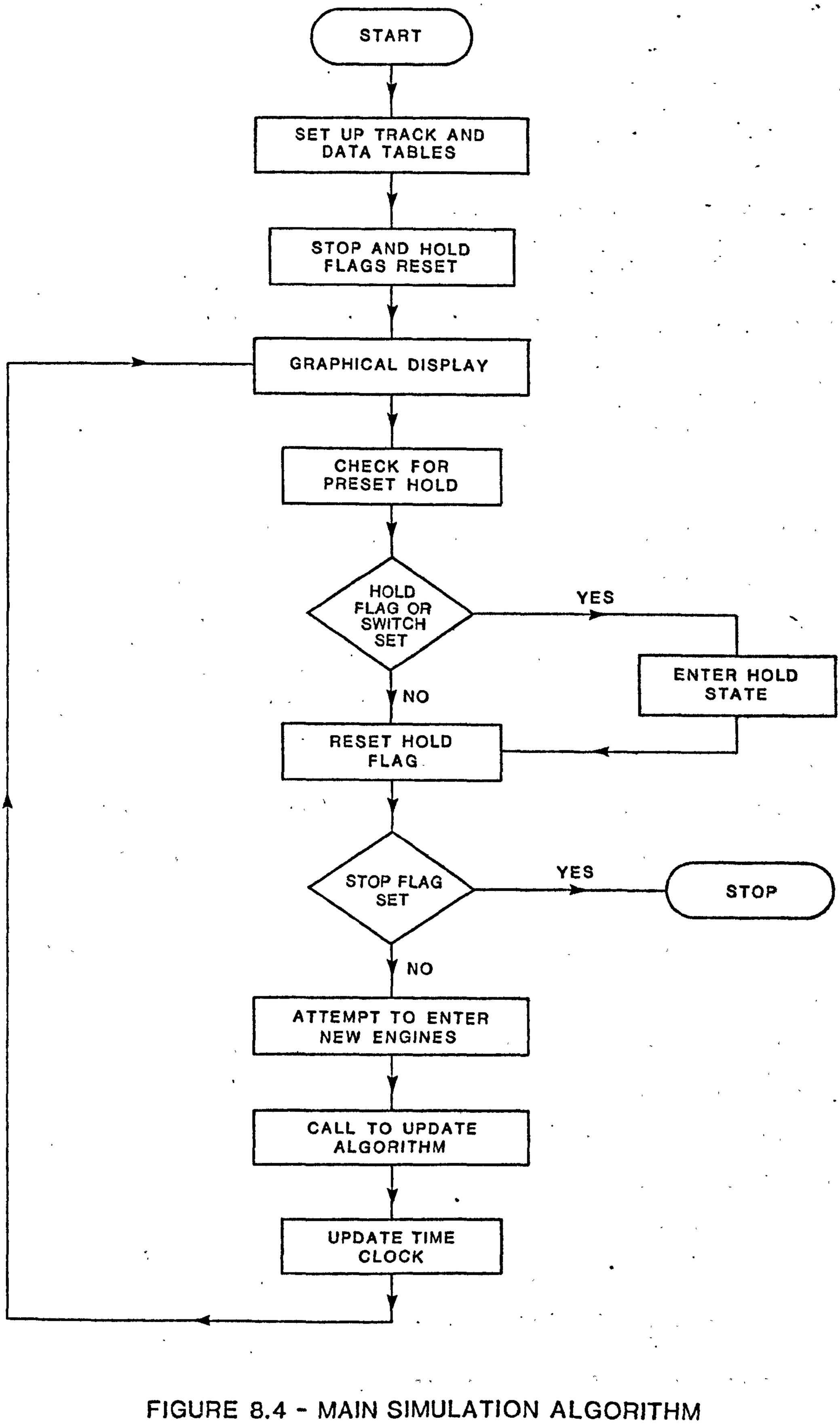


FIGURE 8.4 - MAIN SIMULATION ALGORITHM

1) The total number of track sections in the layout to be simulated.

2) A list of all the data fields for all the required track sections as specified in section 8.3.3.

3) The maximum number of engines to be allowed in the system at any one time.

4) The update time interval. Being a digital simulation the process is not continuous but must occur in discrete steps. The time given for this data item is the interval of each step in seconds. It can be defined as the amount of time the handling takes to move an engine the actual distance regarded as a single unit of length of an ordinary track section. The actual timing structure is thus up to the user, but there are some limitations. As the length of a track section must be defined as an integer of value one or greater, then this puts a limit on the upper value that this interval can take. For instance if we have a handling system that can move an engine at $1\ ms^{-1}$ and there are sections of track only 30 m in length, then the maximum value we can assign to the update time interval is 30 seconds. There is no limit to the minimum time interval other than that it must be greater than or equal to 1 second. However the shorter the time interval, the greater the amount of processing time needed to simulate a given period of real-time and so normally a value would be chosen which provides the largest possible interval consistent with the required accuracy of simulation.

5) A series of data items comprising of various timing

references and indexing values for special track sections.

When this is completed the start-up routine automatically makes a call to the special action control routine with a flag value of + 1. The routine thus called is the one that is used to set-up the statistical control fields described later on. After this has been done the actual simulation can commence.

The first part of the loop representing a single update is the graphical display system. The simulation has the ability to output a graphical representation of its current situation using the Tektronix 4010 visual Display Unit connected to the computer. This facility allows the user to observe clearly and easily the functioning of the system to improve his interaction with it. The whole action of graphical presentation is optional and its presence or absence can be controlled using the computer console SENSE switches.

The interval between successive plots can be controlled. Obviously it would normally be unnecessarily time consuming to plot the layout after every single time increment. Instead the user can specify a parameter in the initialisation data that is to be the interval in terms of the number of updates between successive plots.

It is this graphical display operation that uses the last two data parameters given for all track sections except dummy types. These form a pair of X - Y coordinates relating to positions on the VDU screen. The track layout is drawn on the screen as a series of straight lines by linking the data given for the start of an ordinary track section to the start of the next section. If the listed

next section is a dummy section (a divergency or convergency) then a search is made ahead to find the next real part of the track. Where a holding area, exit or an entry occurs, a small square is drawn around the coordinates specified for that track section. Dummy sections are ignored as they have no physical magnitude (see Figure 8.5).

The exact position of each engine currently in the sytem is marked by a cross on the track position it is at present occupying with engines in holding areas (i.e. under test) marked by a slightly larger double cross. The current simulation system time is also written in one corner of the screen.

After it has, if required, performed the graphical display, the simulation moves on to the part of its operation concerned with the HOLD facility. This is the method by which the user can temporarily halt the simulation and get access to its various functions, thus allowing the interactive function. The different options are described fully in a later section. There are two ways in which a HOLD state can be entered. During any HOLD state the user can specify some time (in terms of simulation time) when the next HOLD is to occur. When the simulation attains this time the HOLD state is automatically entered and operator action requested. Alternatively the user can force a HOLD state to occur during the current update cycle by depressing one of the special SENSE switches on the computer at any time. The HOLD state also provides the method for termination of the simulation by setting an internal flag that the main program checks each cycle and if required stops the execution.

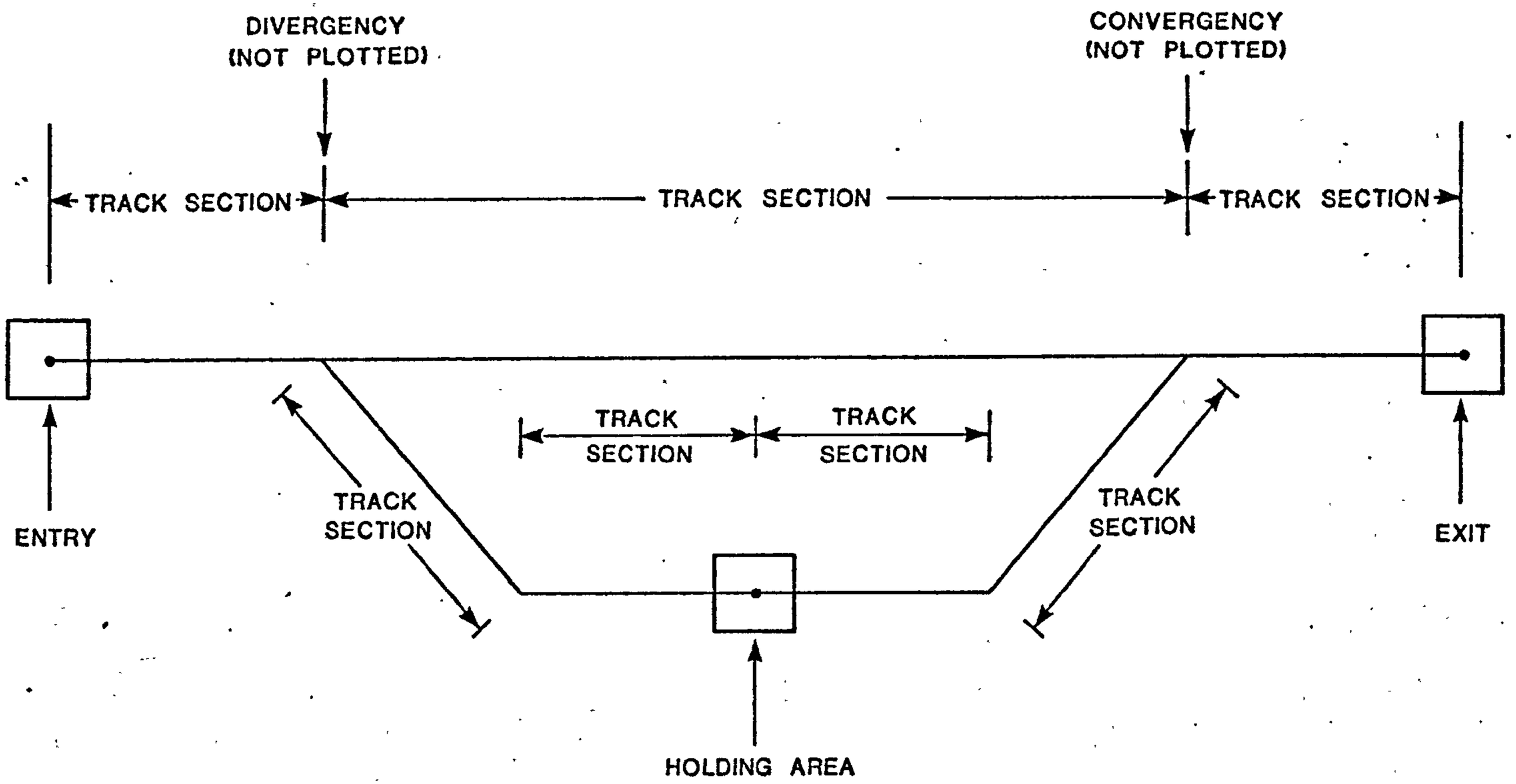


FIGURE 8.5 - V.D.U. PLOTTING TECHNIQUE

As has been mentioned earlier, sections of track designated as entries are processed differently from all other types. The next step in the simulation is to examine all the entry points in turn and to see if they can validly enter a new engine into the system.

When this has been completed the actual update of the system being simulated can be performed. The process works by examining each engine currently listed as being in the system in turn. For an engine in the middle of an ordinary track section the algorithm will search to see if there is another engine in the same track section and in a position one step forward of the current one. If such an engine is found then the next 'space' is already full and the current engine is left where it is. When the space is found to be vacant the current engine data area is altered to move it one forward.

When an engine has progressed to the end of a section, the simulation looks ahead to the next track section. If this is a dummy type, the algorithm continues to predict the path of the engine forward, observing all 'rules of the road' and route decisions, until a real track section is found. If at any stage of the prediction process a situation is discovered which would prevent the transit of an engine through that track section, such as a section with its section blocked flag set, or a convergency with an engine waiting at its entry to the convergency that is its priority entry, then the attempt to move the engine is aborted and it remains where it is.

If on the other hand there is no impediment to the engine progress, the type of the destination section will be checked. If

it is an exit point, the engine is deleted from the data storage buffer with calls being made to the relevant routines that maintain the background statistical information. For a holding area, a check is made to ensure that it is vacant in which case the engine is moved into it. Similarly for a section of ordinary track the first space of the track is checked for occupancy. In both cases an engine will remain in its original position should its planned next position be already occupied. If the move does take place any calls for special action required by the various track sections concerned will be made.

As stated above, normally a special action flag requiring a call on leaving the section is not made until the engine is actually successful in leaving the section. There is, however, one exception to this rule and that is when an engine is about to leave a holding area. In this case any special action call required will be made as soon as an engine is ready to leave the area regardless of whether the attempt was successful or not. This variation in practice was incorporated because of the prime use of holding areas, i.e. to simulate test cells. One of the important statistical parameters involved in evaluating the performance of a particular layout is the percentage of time that a test cell is idle. In an optimum layout this figure should be minimised. The maintenance of the necessary statistical data gathering is performed by special action flags linked to the entrance and exit of each test cell. However if the test has been completed and the engine, although ready and waiting, cannot leave the cell because of some other factor causing a block in the

system, then this represents non-useful cell time.

In general the simulation algorithm will correctly process the simulation of any logically correct layout. Obviously if an impossible section of layout is specified, such as a section sending engines to an entry point, the simulation will fail.

The last action in the loop is simply to update the data items which keep track of the internal timing of the simulation. After this the next update loop will commence.

8.3.4 - Statistical Simulation

One of the most important facilities of the system is to simulate behaviour patterns based on statistical predictions. To illustrate this, let us consider the case of an untested engine entering a test cell and undergoing a test. Now we may know, or be able to predict, the overall percentages of engines which will pass or fail our planned test procedure. However the actual performance of any specific engine will be random in relation to these tested before and after it, other than from the point of view of the overall statistical trends.

This sort of behaviour is catered for in the simulation by a group of subroutines based around pseudo-random numeric operations and normally called as a result of track special action flags.

The heart of all this is a subroutine called RAND that generates a new pseudo-random number every time it is called. The method used is known as the congruence or residue method (57). The basic equation is to give a new pseudo-random number, C_{i+1},

from the previous one, C_i, is:-

$$C_{i+1} = (C_i . \lambda + \mu) \text{ (modulo P)} \qquad (8.1)$$

The true degree of 'randomness' of the numbers produced and the length of the repeat cycle, depends on the values of λ, μ and also C_o, the arbitrarily chosen first random number (called the seed). The range of numbers produced will depend on P and everything can also be dependent on the size of the digital word of the computer used. Although there are plenty of references available for well proven values for large word length machines, some experimentation was necessary to find values which worked well on our sixteen bit machine. The technique finally chosen was based on the Moshman multiplicative congruence values:-

$$\lambda = 7^{17}, \quad \mu = 0$$

$$\text{and} \quad C_o = 17$$

The residue was obtained using a value of P = 1599. This then provided us with pseudo-random numbers in the range 0 → 1599.

The percentage biased results are generated by using a random look-up of a biased field. The initalisation section action call can be used to set up a series of statistical fields, each of which represents a pass/fail decision. The pass rate is specified as a percentage. Internally these fields take the form of 100 word data areas. The 1600 digital bits of each field (16 bits per word times

100 words) are set either to logical 1 (pass) or logical zero (fail) in accordance with the pass rate specified for that field.

During the operation of a simulation the random but percentage - biased result may be obtained by call to subroutine TEST (N, RESULT). The integer N indicates the statistical field concerned and the result of the test is returned as the logical variable RESULT. The technique used is that a pseudo-random number is generated and used to index the equivalent bit in the data field. The logical value of this bit is used to generate the result. As the indexing number is random the sequence of results will be random but the universal distribution of random numbers together with the percentage of true and false bits in the field will over, a large enough sample, result in the correct statistical performance. In this way any event that can be represented in terms of a probability of two alternative outcomes can be simulated. Where more complex alternatives are possible, these must be broken down into a series of logical decisions.

It is also possible to use the subroutine RAND to generate numbers for other purposes and its range can be altered simply by using the equation below:-

$$B = \frac{A \cdot (\text{Result of Call to RAND})}{1599} \qquad (8.2)$$

where B will then be a pseudo-random number in the range 0 → A. The use of this facility is demonstrated in allocating the testing time of failed engines in the example later on in this chapter.

8.3.5 - Statistical Data Collection

As has already been mentioned, one of the most important features of the simulation is the maintenance of statistical data collection routines as a background feature all the time the simulation is running. It is this data that provides the prime means by which a user may evaluate the performance of his system design. All access to the information is performed on an interactive basis by way of the simulation HOLD facility.

The data types collected are:-

i) System Throughput - each time an engine leaves via an exit the throughput via that exit is updated. The results can be obtained either for each exit point separately or for all together to give the total system throughput rate.

ii) Time in System - it can also be informative to know the average amount of time an engine remains in the system as this is a measure of the amount of stock tied up. A very high percentage utilisation of the test cells can be obtained by flooding the track to capacity with engines, but this may result in a large amount of capital tied up in stock-in-hand. Such a situation will show up as a high average time in system. This data can also be taken for the system as a whole or split up for individual exit points.

iii) Holding Area Idle Time - the values of the amount of time a holding area is idle and the percentage of total time that this represents is maintained for each holding area. In their

major usage as test cell simulators, these areas will represent the prime bottlenecks and also major cost contributors in a proposed system. Obviously one of the major objectives of a layout designer will therefore be to obtain maximum utilisation of his test cells. The idle time count provides the necessary performance information.

8.3.6 Run Time Options

Although the prime method of user interaction is based on the use of the hold state, there is an additional facility also available. The EAL 640 has a series of console sense switches. The position of these switches can be examined by a program to generate 8 true/false decisions. The actions of the switches (designated A to H) are given below:-

A) Unless this switch is on the graphical display function is aborted.

B) & C) Not used

D) When on, will cause the simulation to enter a hold state during the next update cycle.

E) Every time a new engine enters the system the current simulation time, engine reference number and actual entry point will be printed on the teletype if this switch is on.

F) Has the same function as E) above except for exit points. In addition the status of the engine leaving the system is also printed.

G) When on, a message is printed each time a new 'day' in simulation time begins. If supressed the day as well as time infor-

mation is printed for E) and F) above.

H) This switch disables all the entry points to the system regardless of blocks, etc. and can be used to clear the system prior to a period of simulation.

Obviously these functions are dependent on access to a set of console sense of data switches. If these are not available the control would have to be by way of Hold state commands.

8.3.7 - Interactive Hold State

If at any time the simulation requires the entry to the HOLD state, either due to depression of sense switch D, or as a result of a pre-set time elapsing then processing of the simulation is temporarily suspended and the HOLD state entered. In this mode the user is provided with a variety of interactive functions as a result of an exchange of dialogue using the computer teletype printer and keyboard.

When the HOLD state is entered the computer types a message asking the user to enter a code number designating which of the various functions available he requires. The different functions are described below, listed under the code numbers that identify them:-

zero) STOP - set the system stop flag so that on completion of the present HOLD state the simulation will be terminated.

1) PUNCH - will cause the computer to punch out a data tape of the current simulation track data, etc. which can be used during a subsequent simulation start-up without requiring the off-line

punching of a new data tape to allow the inclusion of any interactive modifications to the system.

2) BLOCK - the user will subsequently enter the number of a track section (the computer will check for validity) and the section blocked flag is then arbitarily set. This can be used to simulate the failure of a track section.

3) UNBLOCK - same as BLOCK except that the relevant flag is reset.

4) STATISTICAL INTERFACE - this is the way in which the user gains access to the background statistical data. The computer first asks which of the three types of data (see section 8.3.5) the user is interested in and then the particular sub-section involved (i.e. which exit point or holding area). The user is also asked if he requires the data section to be reset. If he signifies that he does, after printing out its present contents the data area is reset so that data collection is restarted from the current simulation time. Alternatively the statistical data will continue to be collected cumulatively.

In all cases the current contents of the relevant statistical area are printed out. The information presented for each type is as follows:-

i) System Throughput

   a) Exit Number

   b) Equivalent Track Section

c) Present Simulation Time

d) Simulation Time when last reset

e) Total number of engines throughput since last reset

f) Average throughput per hour since last reset

ii) Time in System

a), b) and c) As for i) above

d) Cumulative total time in system for all units leaving via this exit since last reset

e) Total number of engines throughput since last reset.

f) Average time in system per engine since last reset.

iii) Holding Area Idle Time

a) Holding Area Number

b) Equivalent Track Section

c) Present Simulation Time

d) Total idle time since last reset

e) Total number of engines throughput since last reset.

f) Average idle time per engine throughput.

g) Percentage of total time spent in idle condition.

Using the data obtained, and the various track modification facilities described later, it is possible for the user to perform interactive optimisation of his design.

5) SPECIAL ACTION FLAG - after obtaining a track section number from the user, the computer alters that section's special action flag to the new value specified.

6) ALTER TRACK DATA - the computer asks for a track section number and data item number (corresponding to Figure 8.2). The number subsequently specified is used to replace the data item of the track section already defined. It is by the use of this command that the user can make any alteration he wishes to the track layout during the course of the simulation.

7) ALTER ENGINE DATA - in the same way that 6) above alters any item within a track section data area, this command can be used to modify any data item representing an engine.

8) SET HOLD TIME - this command is used to define the next time (in terms of simulation time) that the hold state is to be entered. This function is complementary to, rather than an alternative to, the HOLD call due to depressing the relevant sense switch.

9) CONTINUE SIMULATION - causes an exit from the HOLD state to allow the simulation to continue.

After each action has been completed, the computer will request the command for a new HOLD function. In this way as many different HOLD functions can be called during a single HOLD state as required until terminated by the CONTINUE SIMULATION command.

8.4 Test Facility Example

Having seen what the simulation package consists of and how it works, we can turn to an actual example of how the simulation can be used to represent a given layout and to investigate its performance.

In a production test environment we will often be using the concept of several small parallel systems rather than a single huge system, because in this manner the effect of a system breakdown is reduced. The example we shall consider here is of a sub-section consisting of 8 engine test beds as shown in Figure 8.6. Engines arrive from the production line and are to be sent to one of the test beds. The engines which pass the test will be despatched to the finishing line and shipping department.

Whilst the passed engines must obviously remain on the test bed until the total test procedure is complete, engines which fail will be removed as soon as they fail any part of the test procedure. These failed engines fall into two categories, those which can be repaired and those which cannot and must be sent back to the production area for rebuilding. There is a repair area for the first category and these are subsequently returned to the system for retesting.

8.4.1 - Layout Simulation

The first stage of the simulation process is to layout a preliminary track system as shown in Figure 8.7. Furthermore preliminary values for all other variables should be chosen. The actual values used are not critical as they can be altered interactively at any time.

Initially let us make the following stipulations:-

i) The track speed will be 40 feet per minute.

ii) The full testing procedure will last 30 minutes.

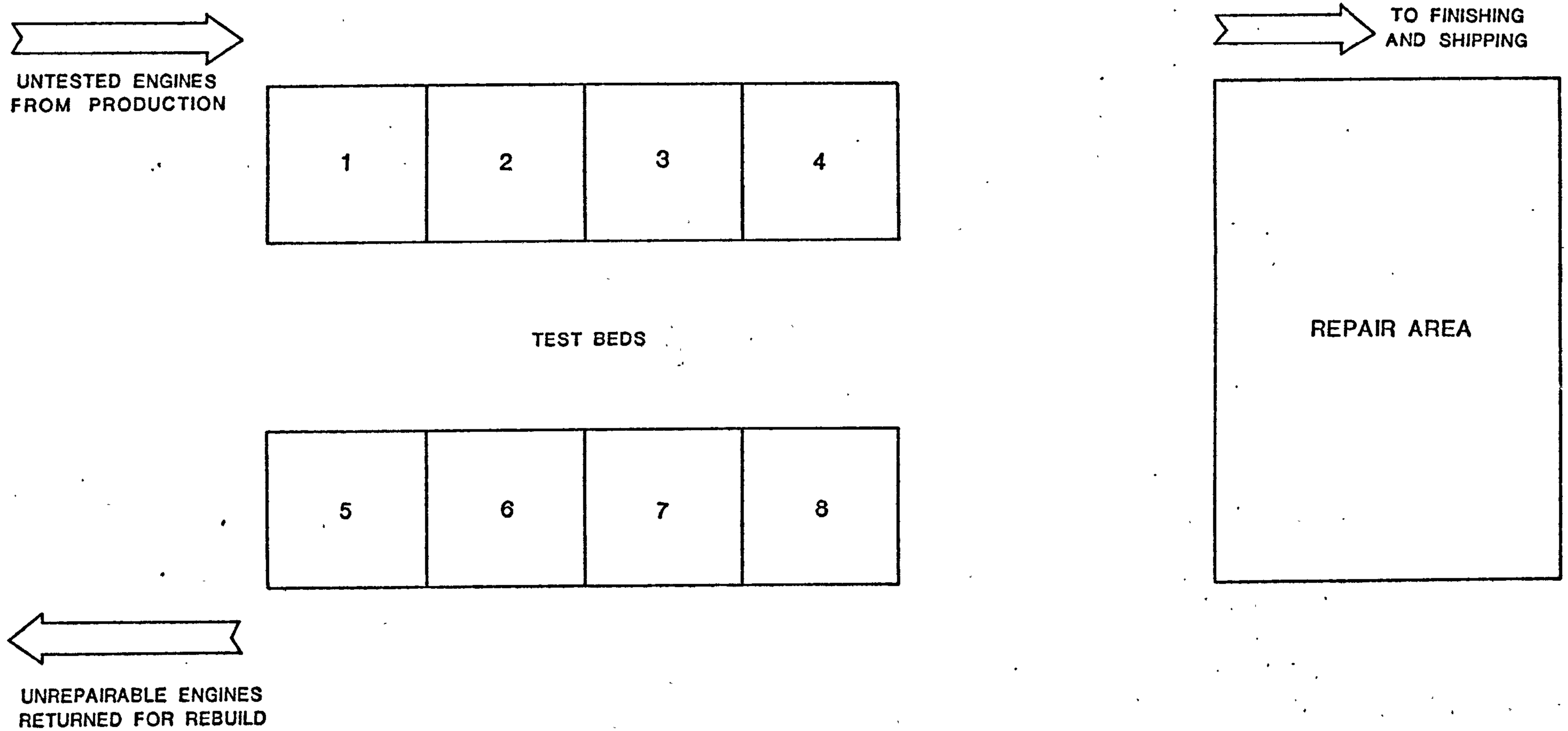


FIGURE 8.6 - BASIC LAYOUT

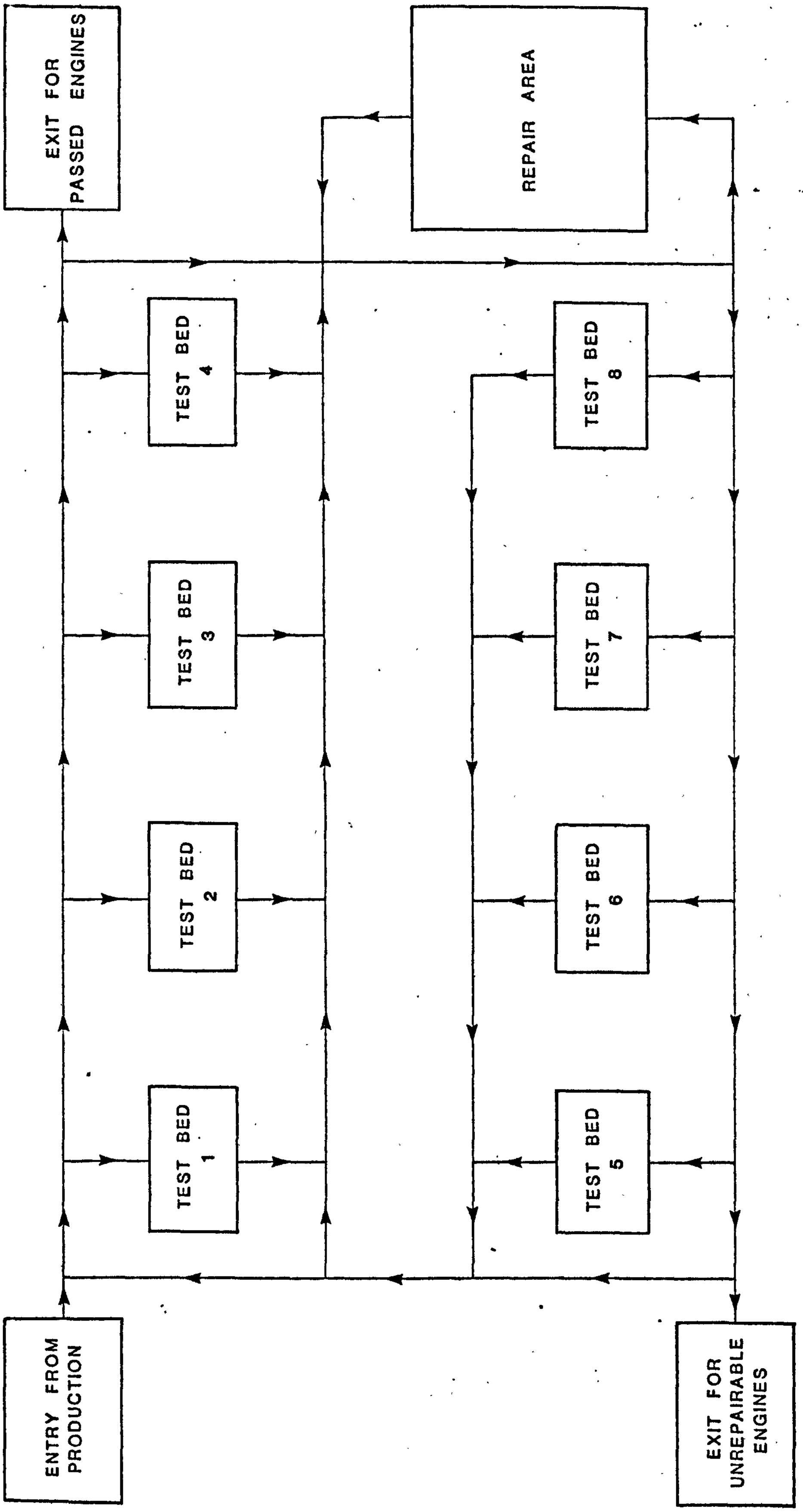


FIGURE 8.7 - TRACK LÁYOUT

Those engines failing the test will do so randomly during any point between the start and the finish of the normal test procedure.

iii) The repair area can accommodate no more than five engines at a time. The duration of the repair time is assumed to be 15 minutes.

iv) The spur leading into each of the eight test beds will be controlled specially. The divergency from the main loop will divert untested engines towards the test bed until two have passed. It will then refuse to admit any more to the spur until one of the engines has finished its testing. In this way the normal state of affairs will have an engine on the test bed with another untested engine waiting.

v) Rather than allow the system to fill up until it is full of engines, the entry point will only permit a maximum of 16 untested engines or engines under test to be present in the system at any one time.

vi) We will define five different possible numbers to represent the status of any given engine. These are:-

Status = 1, Untested Engine

Status = 2, Passed Engine

Status = 3, Failed Engine but repairable

Status = 4, Irrepairably failed engine

Status = 5, Engine under repair.

The next stage is to draw out our layout and break it down into the different track sections which will make up the track data

fields for our simulation. This is shown in Figure 8.8.

Before starting to work out the various data items for each section of track we must pick the interval in simulation time between updates. The value chosen is 15 seconds which means with a track speed of 40 feet per minute, each update 'moves' an engine 10 feet. As far as the lengths of the ordinary track sections are concerned, for most of them the selection criteria will be the time it takes an engine to traverse them rather than the number of engines they can actually contain at any one time. This is because the limitation in the number of engines in the system will result in unblocked movement of an engine through most track sections. Thus if, for example, a track section is really 60 feet in length, its data item value will be 6 as given by six 'steps' of 10 feet.

Figure 8.8 shows how we will use a total of 94 track sections to simulate the layout. It is not proposed to go through how the entire data field is constructed but rather to examine several sections which are generally representative together with some areas of special interest.

The point where untested engines appear in the simulation must be an entry point and is designated as section 1. This is followed by a section of ordinary track. Where this is joined by the main loop we must have a convergency (numbered 3). Another section of ordinary track leads to the turn off point of the first test bed. Engines may either continue along the loop (section 6) or be diverted by way of track sections 54 and 55 to the test bed itself, section 56.

The data fields used to simulate this part of the track are

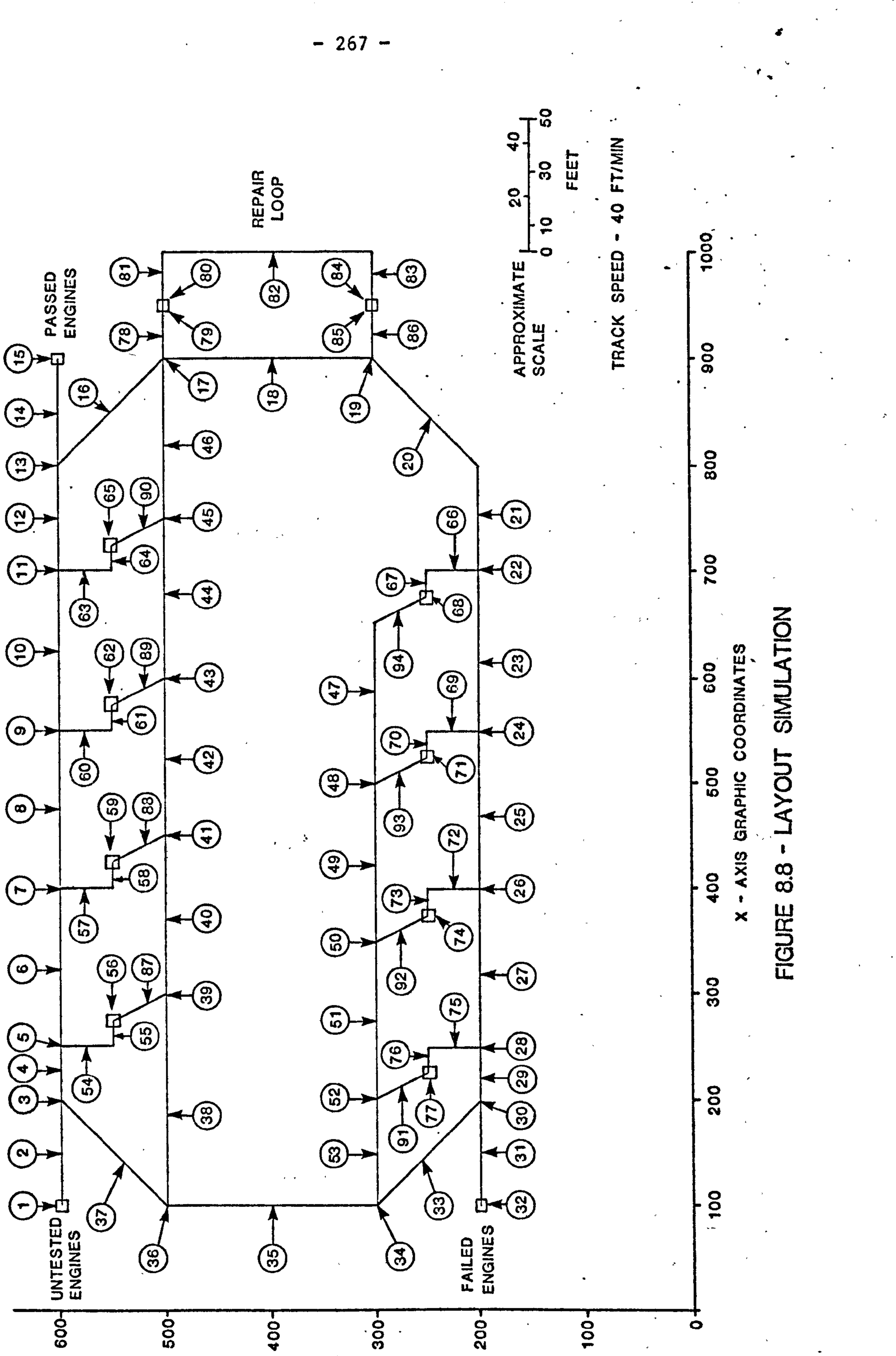


FIGURE 8.8 - LAYOUT SIMULATION

shown in Figure 8.9. The reasoning for the data fields is set out below:-

i) Section 1 : Following Track Section = 2

Type = Entry = 6

Item 3 has no function

A special action flag will be needed to close the entry point when the maximum permitted number of untested or under test engines (16) are in the system. This is done by a call to the second special action routine on leaving this section (Flag = - 2) which then sets the block flag for the next section.

Entry Status = Untested = 1

Graphical Location Coordinates = 100,600

ii) Section 2: Following Track Section = 3

Type = Ordinary Track = 1

Length; except in special circumstances engines will move freely along the track. Actual length is 40 feet. Time taken in transit at 40 feet/min will be 1 minute which equals four fifteen second update intervals thus length = 4.

Special Action Flag - none = 0

Section Blocked Flag - initially unblocked = 0

Coordinates of start of section = 100,600

TRACK SECTION	DATA ITEM						
	1	2	3	4	5	6	7
1	2	6	X	-2	1	100	600
2	3	1	4	0	0	100	600
3	4	3	X	37	0	X	X
4	5	1	2	0	0	200	600
5	6	2	1	54	0	X	X
6	7	1	6	0	0	250	600
54	55	1	1	3	0	250	600
55	56	1	1	-4	0	250	550
56	87	4	120	-5	0	275	550
13	16	2	2	14	0	X	X
19	20	2	3	86	0	X	X
30	33	2	4	31	0	X	X
36	37	2	4	38	0	X	X
17	18	3	X	16	0	X	X
34	35	3	X	33	0	X	X

X = DON'T CARE

FIGURE 8.9 - TRACK DATA

iii) Section 3: Following Track Section = 4

Type = Convergency = 3

Item 3 has no function

Priority Previous Track = Main loop = 37

Section Blocked Flag - unblocked = 0

Items 6 and 7 have no function

iv) Section 4: Ordinary Track Section 20 feet long,

Thus length = 2.

v) Section 5: Priority Following Track Section = 6

Type = Divergency = 2

Diversion Status = Untested = 1

Secondary Following Track Section = 54

Section Blocked Flag = Initially Unblocked

= 0.

Items 6 and 7 have no function

vi) Section 6: Ordinary track section 60 feet long, thus

length = 6.

vii) Sections 54 and 55: Two ordinary track sections forming the spur leading to the first test bed. The reason for using two sections is that two different special action calls are required. Firstly when an engine enters the spur, a check must be made to see if the spur should be blocked off as it holds the

additional untested engine as well as there being an engine under test on the bed. This is accomplished by a call to the third special action routine whenever an engine enters section 54 (special action flag = 3). The second special action call is made when an untested engine enters the test bed. The test bed idle time count is halted, a call is made to the TEST routine to identify whether or not the engine will pass the test. If it will fail a second call is made to find out if the cause of the failure is repairable or not. The relevant new status is entered in the engine's data area. The time duration of the engines stay on the test bed is also defined at this stage. The update algorithm will have already entered the pre-set test duration into the engine's data area. However, if the engine is going to fail, this value is altered by a call to the random number generating routine to modify the test duration so that it will be anything between zero and the full duration. All this work is accomplished by a call to the fourth special action routines whenever an engine leaves section 55 (special action flag = -4).

As the transit time for the spur will tend to depend on what is going on in other sections rather than the actual length of the track sections and the track speed, we can assign minimum value of 1 to the simu-

lated lengths of the two sections.

viii) Section 56: Following Track Section = 87.

Type = Holding Area = 4

Holding Time = 30 minutes (unless altered by special action call) = 120 updates.

On completion of the test when the time count reaches zero, the idle time count for the test bed will be resumed. The system entry point must be allowed to enter a new engine (i.e. track section 2 is unblocked) and the entrance to the test bed spur also unblocked to permit another untested engine to enter. This is accomplished by the call resulting from the special action flag of - 5.

Most of the remainder of the layout is simply a repetition of the sub-section described above. The special action flags are generalised, that is the flags are the same for all equivalent test bed spur sections with the routines themselves being responsible for blocking or unblocking the correct track sections.

Engines leave the main circuit by being switched at one of the divergencies numbered as sections 13, 19 and 30. Figure 8.9 shows these sections have the main loop as their priority following section with the relevant diversion status for their spur lines. There is also a divergency labelled section 36 which allows unrepairably failed engines to take a short cut avoiding a section of

the main loop on their way to the exit point at section 32. Where the engines returning from test beds rejoin the main loop there are convergencies (sections 17 and 34). Section 17 in fact has 3 track sections sending to it (16, 46 and 78), the additional one being the return from the repair area loop. Both convergencies have the relevant main loop track section as their priority sending track.

The simulation of the repair area is not quite so straightforward. As Figure 8.10 shows it is accomplished by what is really a totally separate layout connected to the main system only by the behaviour of certain special action routines. The reason behind this is to allow the collection of data on engines being repaired by having them pass through an exit point. We can see how the linkage works if we examine Figure 8.11. Initially the entry of units to the repair loop is stopped by the action of the section blocked flag of section 83. However when an engine leaves the system via section 85, one of the special action routines is called. This unblocks section 83, thus resulting in the creation of a new engine by section 84. However this action in turn recalls the same routine which will then reblock section 83. In this way, every time an engine disappears via the exit, a new engine will be created by the entry to the repair loop. Exactly the same procedure is used to link sections 79 and 80 in order to simulate engines leaving the repair loop and returning to the main layout to be retested.

The duration of repair, 15 minutes, is simulated by making section 82 a track section of length 60. However this would also allow it to contain 60 engines under our basic algorithm and we have

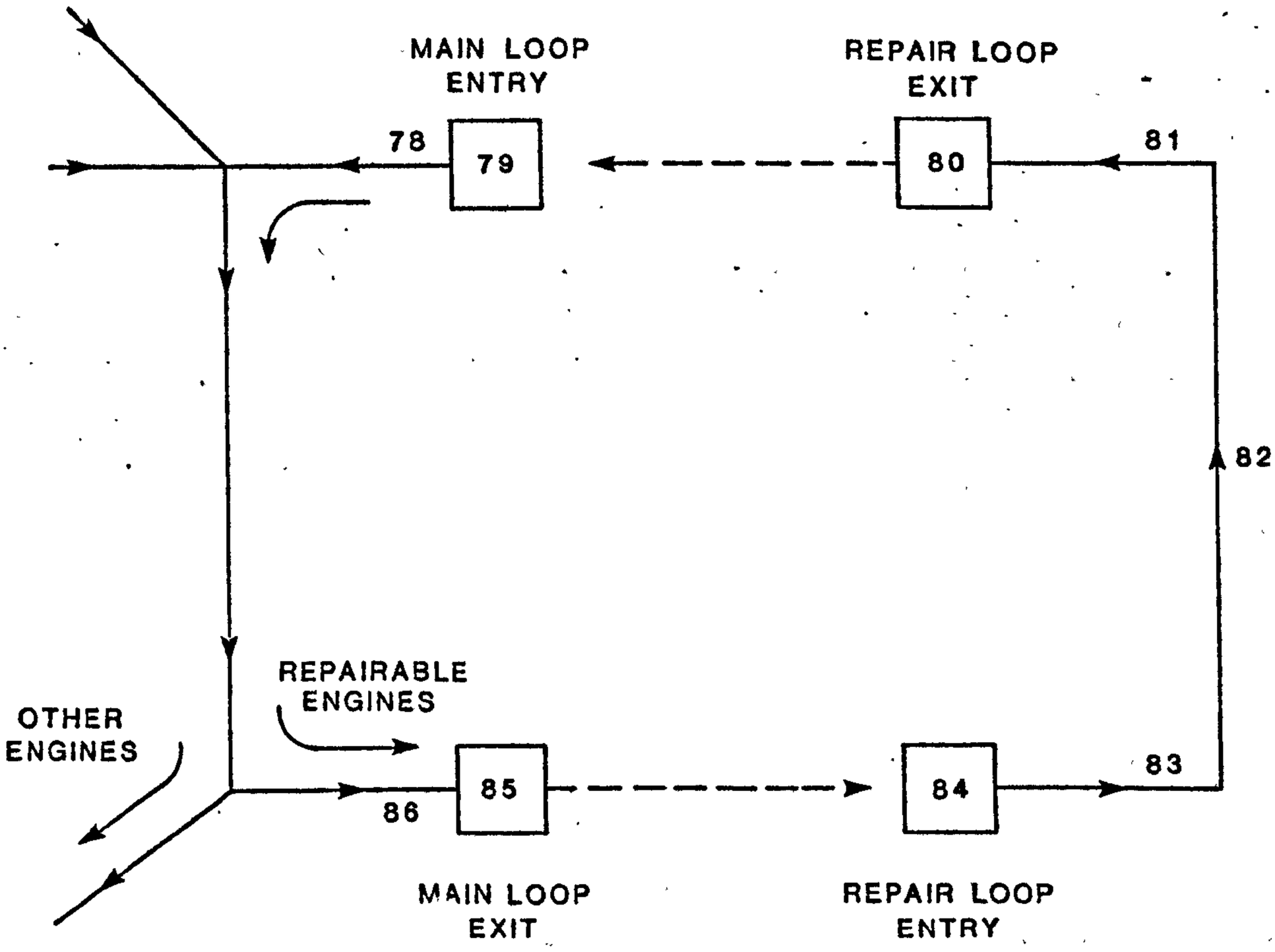


FIGURE 8.10 - REPAIR LOOP SIMULATION

DATA ITEM / TRACK SECTION	1	2	3	4	5	6	7
78	17	1	2	0	1	950	500
79	78	6	×	-6	1	950	500
80	×	5	3	6	0	950	500
81	80	1	1	0	0	1000	500
82	81	1	60	-8	0	1000	300
83	82	1	1	0	1	950	300
84	83	6	×	-7	5	950	300
85	×	5	4	7	0	950	300
86	85	1	2	8	0	900	300

FIGURE 8.11 - REPAIR LOOP DATA

stated that the maximum to be allowed is 5. To allow this to operate we have an additional special action routine called by entry into section 86 and exit from section 82. Whenever an engine enters section 86 a check will be made to see if there are now five engines in the repair area and if so the section will set its own block flag. When an engine completes its repair and leaves section 82, the special action routine will unblock section 86 to allow the entry of another engine.

8.4.2 - Simulation Results

Once the layout data has been assembled the simulation can commence. To illustrate the use of the interactive capabilities of the simulation, some details of a test run using the above layout are described.

After loading the data tape defining the layout structure, the computer asks for the various statistical field percentage settings. In this case the values requested were:-

A) 72% of engines taking the test will pass.

B) Of those failing, 41% will have faults that can be dealt with in the repair area.

After this has been entered the time simulation was commenced. The plotting of the layout on the V.D.U. can be seen in Figures 8.12 and 8.13 with the crosses marking the engines moving through the system. The first stage was to allow a period of time to pass to

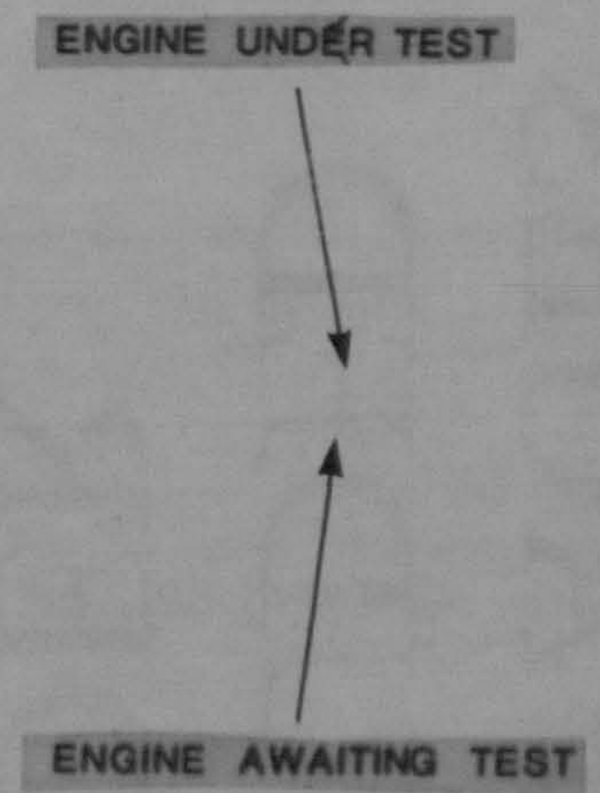
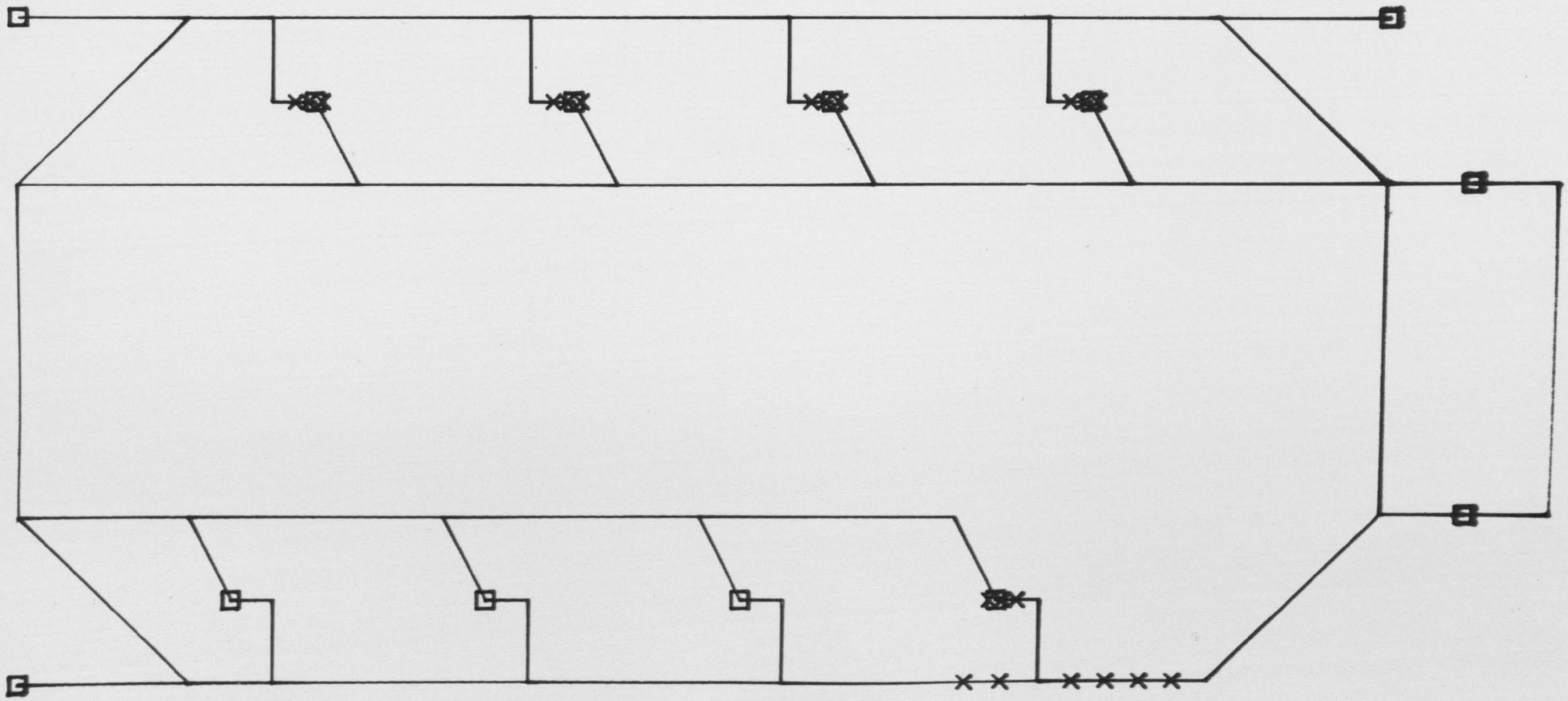


FIGURE 8.12 - V.D.U. OUTPUT LAYOUT (1)



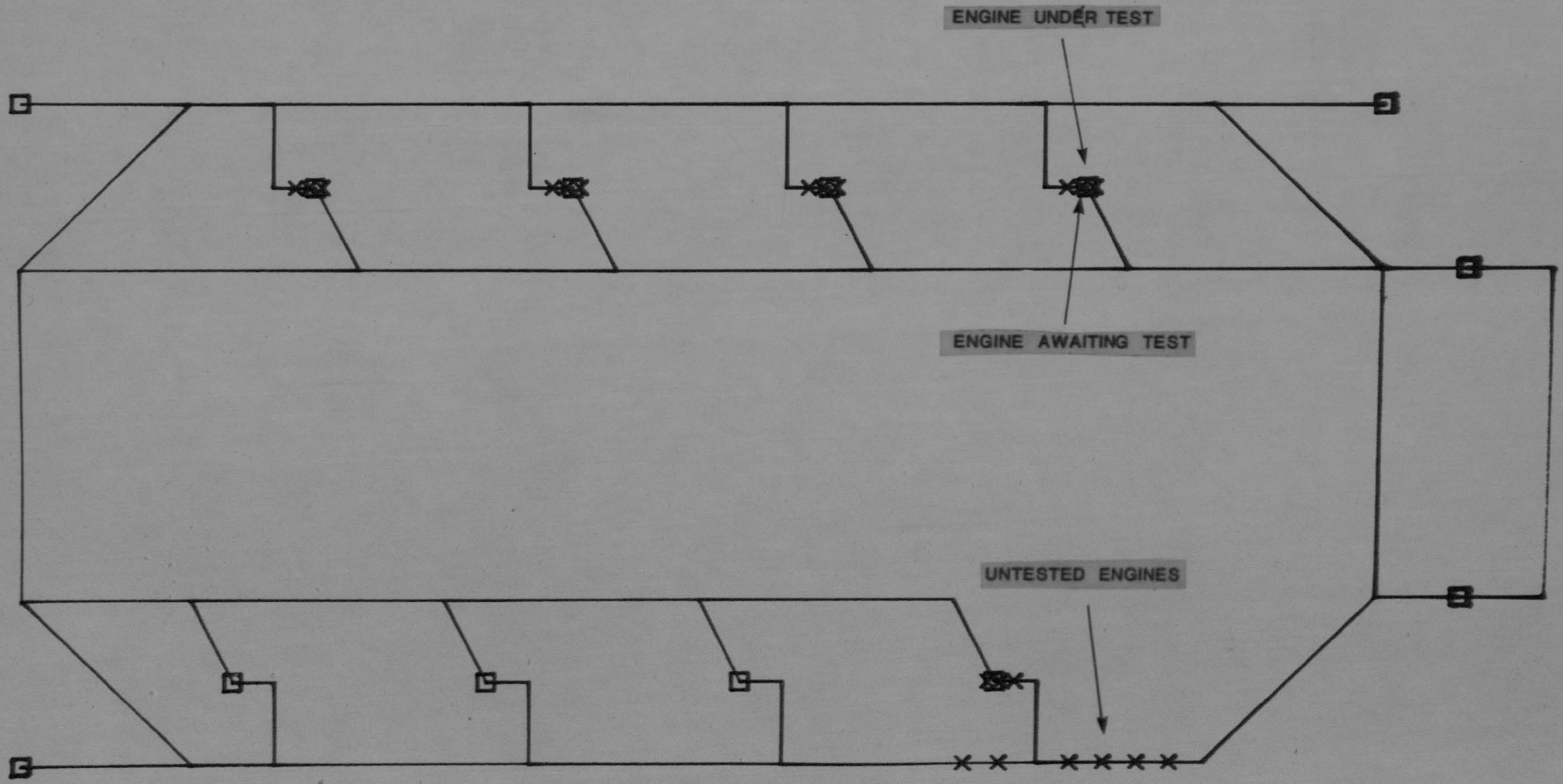


FIGURE 8.12 - V.D.U. OUTPUT LAYOUT (1)

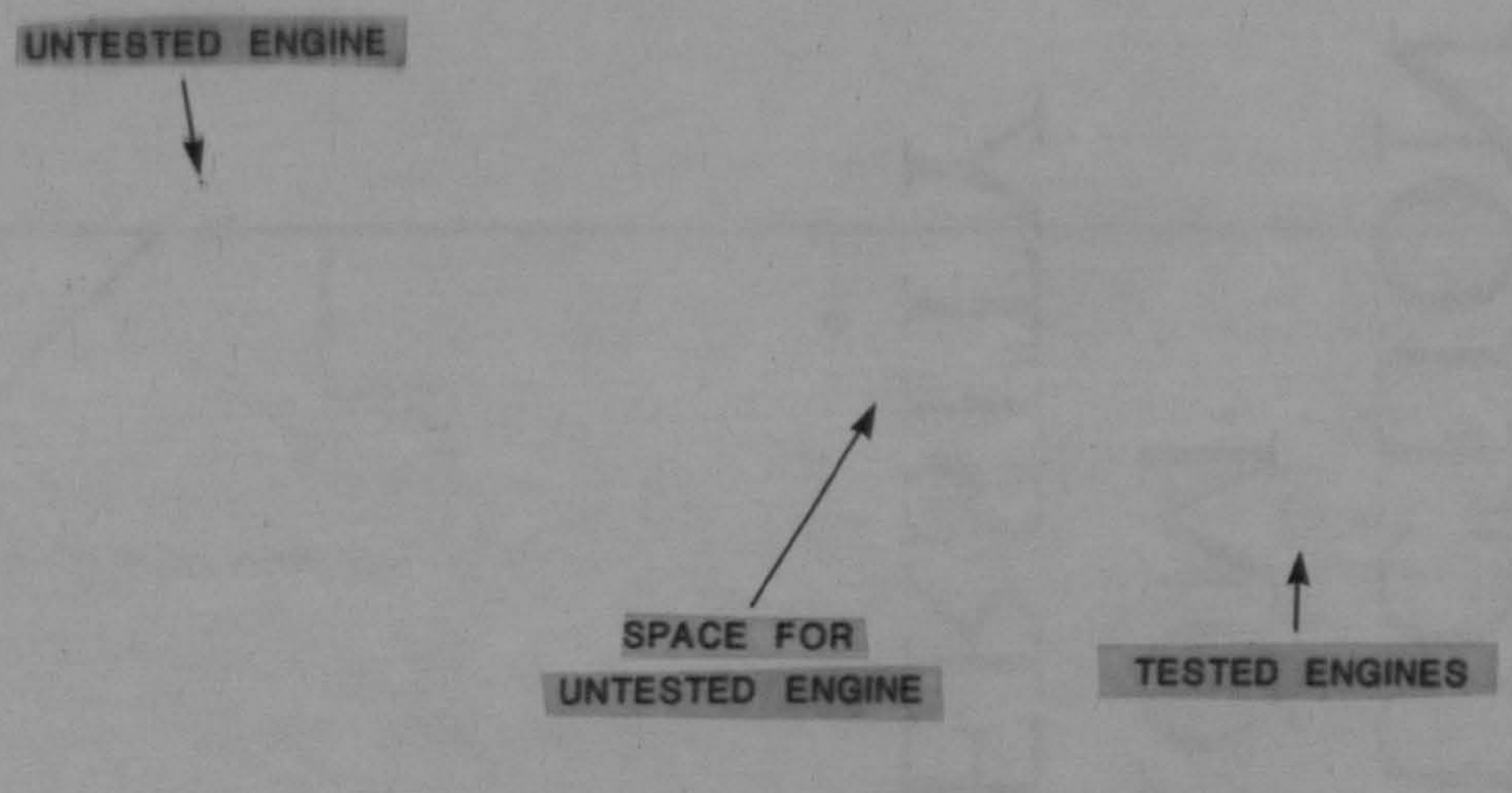
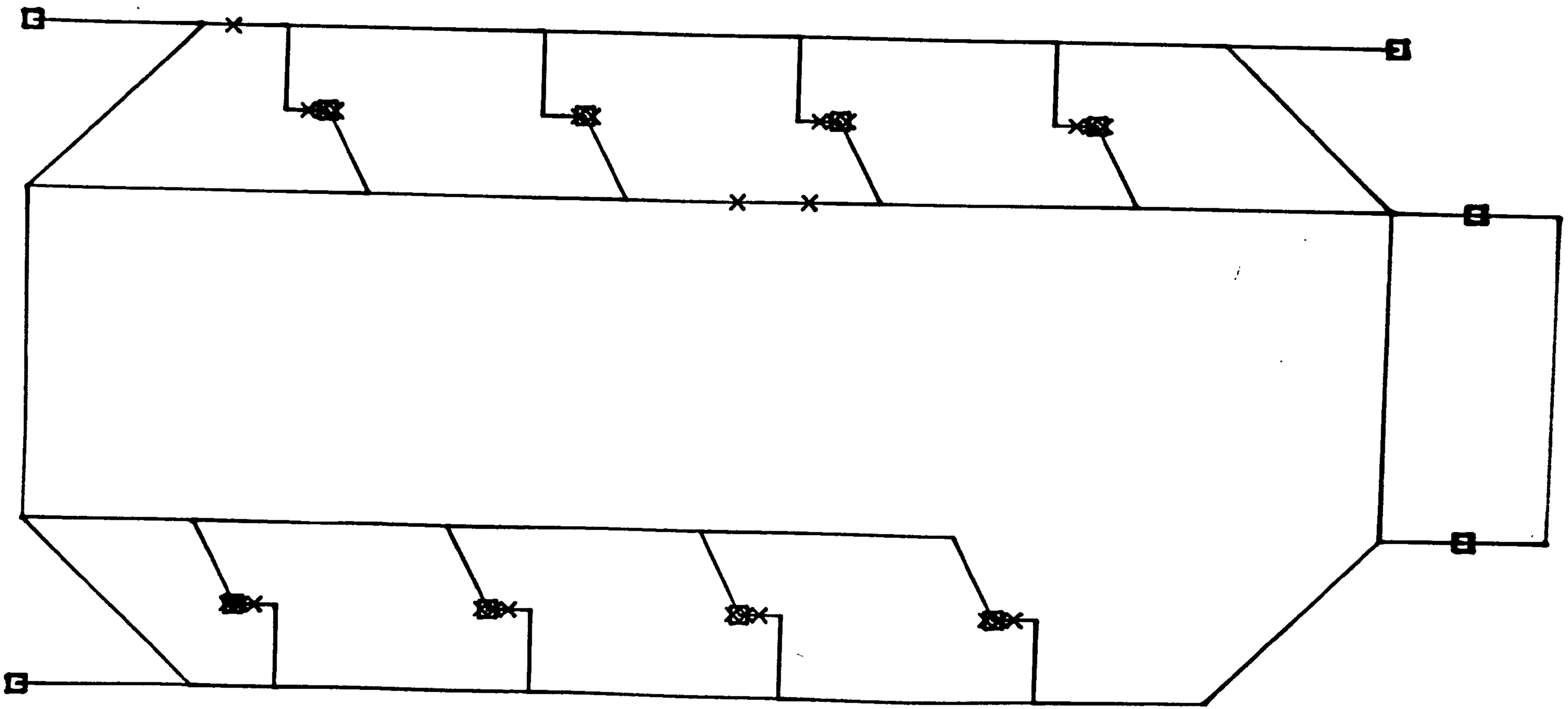


FIGURE 8.13 - V.D.U. OUTPUT LAYOUT (2)



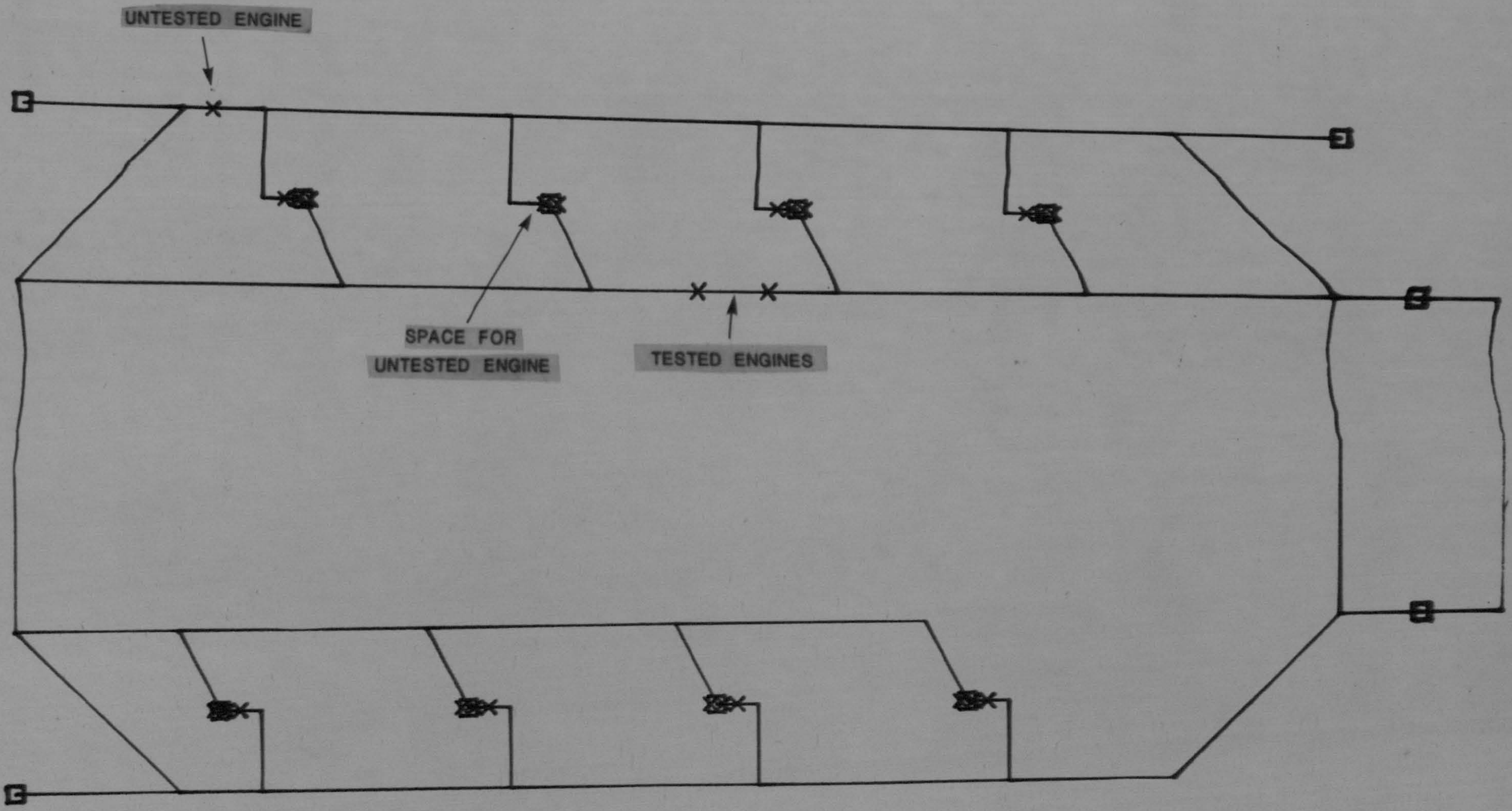


FIGURE 8.13 - V.D.U. OUTPUT LAYOUT (2)

collect statistical data on the performance of the layout before any changes were made so as to provide a baseline against which the effect of any modifications could be assessed. The data was collected for a total of 24 hours of simulation time. To examine the variability of results the throughput out of the passed engines exit was recorded every 8 hours. The results obtained for this exit point were as follows:-

	Total Throughput	Engines/Hour
1st Period	95	11.875
2nd Period	94	11.750
3rd Period	91	11.375
TOTAL	280 AVERAGE	11.667

Maximum variation from average = 2.5%

The figure quantifying the variation will help in saying which changes in statistical data are due to random fluctuations and which are actually significant in terms of any alteration made to the layout.

The first test was to use the HOLD facility to place a permament block on track section 38 to see if this short cut had any real effect on the system performance. If so, it was expected that this would show up primarily as a reduction in the time in system data for the failed engines exit. However when the simulation was allowed to continue for another 24 hours with the block in place the following results were obtained:-

	NORMAL	WITH BLOCK
i) Passed Engines		
Units/Hour	11.667	11.667
Average Time in System (hours)	1.4321	1.4200
ii) Failed (unrepairable) Engines		
Units/Hour	2.708	2.708
Average Time in System (Hours)	1.4425	1.4266

Throughput unchanged

Average time in system decreased by 0.84% for passed engines and by 1.10% for failed engines.

So in fact we can see that, if anything, the removal of section 38 from the layout might well improve the performance. This is possibly due to the fact that the convergency at section 17 gives priority to the main loop, but the variation is so small as to make it difficult to be certain if this is a genuine improvement.

The next stage of the experiment, having unblocked section 38, was to examine the effect of a malfunction in one of the test beds. To do this the test bed represented by track section 62 was put out of order by placing a block on track section 61. The simulation was then run for a further 24 hour period. At the end of this time the data showed:

	NORMAL	BLOCKED TEST BED
i) Passed Engines		
Units/Hour	11.667	10.25
Average Time in System (hours)	1.4321	1.5825
ii) Failed (unrepairable) Engines		
Units/hour	2.708	2.458
Average Time in System (Hours)	1.4425	1.5887

Thus the resultant changes due to the failure of a single test cell in that position are:-

	Units/Hour	Average T.I.S.
i) Passed Engines	- 12.14%	+ 10.50%
ii) Failed Engines	- 9.23%	+ 10.14%
Average	- 10.69%	+ 10.32%

This shows the size of the effect of the failure is considerable but nevertheless seems to be slightly less than the figure of 12.5% (one in eight) that might at first be expected when considering the failure of one of the eight test beds in the system.

The above exercise shows in a very limited way the possibilities resulting from the use of this simulation. Any number of alterations or situations can be put to the test and data on the systems behaviour obtained easily and efficiently. The actual speed at which the simulation progresses in relation to real time is not

fixed but depends on the work-load on the basis of the number of engines in the system, the complexity of the layout and to a large extent the amount of input/output communication required. For the above exercise the simulation time was between 50 to 100 times faster than real time.

8.5 Summary

The simulation of the handling system to be used in an automated production test facility to allow design optimization is an important part of the overall system design. A simulation has been developed which allows the user to interact with the simulation in an effective manner. To simulate the layout of a handling system within the environment of a digital computing system, a number of seperate simulation entities are used. Each entity describes a basic facet that may be found in a handling system and the combination of these different types of track in the form of a numerical table, allows many different types of handling system layouts to be easily simulated. The simulation can then provide the user with the abilities to modify his layout dynamically during the course of the simulation and provides performance data to enable him to evaluate the behaviour of the system.

Chapter 9

DIAGNOSTIC TECHNIQUES

9.1 Introduction

In the previous testing techniques described so far, one particular area has not been covered in detail and that is the interpretation and use of the results. This is because, for the most part, these functions are divorced from the actual testing. Nevertheless, in recent years, one type of testing in which this final stage forms an integral part of the engine test automation system has increased in importance. This occurs in the production test environment, for when an engine fails the test it is important to know the cause of the failure since this will effect what happens next. In the case of a minor fault, correction may be made on the test stand. More serious malfunctions will require taking the engine away to a repair area, whilst in the worst cases the engine must go to the graveyard to be scrapped.

Obviously the best, and quickest way to do this, is to design diagnostic capabilities into the engine test automation system. Some of these types of tests are described in Chapter 4 and can be seen to differ in some respects from the more conventional test types. What we are concerned with here is not so much the details of how every different fault can be identified but rather to set out in general terms the demands that diagnostic testing will make on the system. It is these requirements which have to be taken into account if our future schemes are going to have the ability to perform in a full spectrum of applications.

Examining the range of diagnostic type tests, two generally prevalent factors become apparent about them when comparing diagnostic

and other tests.

1) The control requirements for diagnostic tests are seldom very critical in comparison to conventional tests. Any system that is capable of the level of control needed to carry out performance and emission tests, particularly those that are dynamic in nature, will not have any problems in performing the control for diagnostic tests.

2) A common feature of many diagnostic tests is the high frequency of data logging required. In the case of other testing methods, data logging tends to consist of a large or medium number of parameters being logged at comparatively long intervals. Diagnostic testing on the other hand often involves a short burst during which a single parameter is repeatedly logged at high frequency. This is often used to obtain the characteristic for that particular engine parameter over one engine cycle.

To investigate the requirements of diagnostic testing and the problems associated with it, the following sections give the details of such a testing package implemented on the system at the College.

9.2 High Speed Sampling

The maximum frequency at which data can be logged in a digital processing system depends on the nature of the data signal. If the signal takes the form of digital information, the only limit is the speed at which the computer can fetch digital words from an input/output

port and transfer them to its memory. However when dealing with most process systems this is slightly misleading as most of the process variables which are likely to be investigated are basically analog in nature and hence what may appear to be a digital signal to the computer is the output of some form of converter and hence the actual rate at which the basic process is being sampled really depends on that converter.

Instead it is probably better to consider the case of an analog voltage signal which is fed to the computer, analog to digital conversion system (ADC). Now in this case the maximum frequency will almost certainly be limited by the speed of operation of the ADC system. Nowadays very fast converters are available but even so the time taken is not ususally as fast as the processor instruction cycle time.

If we turn to considering the data itself, the maximum rate of sampling that might normally be required would be around one reading for each degree of revolution of the engine crankshaft. If we limit ourselves to a maximum speed of around 100 rev/s this gives a frequency of 36000 samples per second, or an interval between samples of about 28 microseconds. This is within the capabilities of many converters at present available.

Another factor which may limit the sampling frequency is the duration of sampling and the maximum amount of storage available in the system. In saying that the time taken to store the data is not critical, the use of the processor's own directly addressed random access memory is assumed. Any attempt to use bulk storage for the

incoming data may well provide much more severe constraints on data logging frequency. Thus unless we are going to have a much larger memory than is usually available in a typical process control system, we are going to be limited to a fairly small data area, a typical figure perhaps being 4 K words.

Although this would not be a problem for sampling just one revolution of the engine, it may be that we require 20 cycles to allow, for instance, digital filtering of the signal to be performed. If this is so our maximum frequency of logging is effectively limited by the need to spread our maximum number of samples over the required duration of logging.

In providing a high speed sampling system using the General Automation SPC-16 at the college, the most serious constraint was provided by the slow speed of the system analog converter and signal multiplexer. The former took about 40 microseconds to perform a conversion. In addition the process interface design was such that the multiplexer had to be reset after each sample despite the fact that only one channel was being used. This added a further 50 microseconds to the sample time giving an overall maximum frequency of about 10 kilohertz.

The software package which was written for high speed sampling consisted of three different routines. The master program called H$SP1 could be called as a subroutine by any program written to perform some form of data acquisition and/or diagnostic analysis. When called, it provided a generalised high speed logging system, which could be tailored to the caller's specific requirements at execution time.

Examining Figure 9.1 we can see that after the initial headings have been printed out on the teletype, the user must first specify which of the 32 computer analog input channels the data signal he wishes to sample is connected to. This is followed by the entry of the total number of sample points to be collected and the start of the area in memory where the resulting array of data is to be stored. At this point the program checks to see if the required number samples will in fact fit into the memory space available for storage. If an overflow condition is detected the operator is informed by a printed message and asked to redefine the necessary parameters.

Ultimately the start of the sampling operation itself will be signalled by the occurrence of a special external interrupt. This can be generated either by the user pressing a push-button or by some piece of electrical hardware on the test rig such as a T.D.C. reference sensor. The operator must define the off-set in time between this interrupt and when he wishes the first sample to occur. The need for some internal processing time overhead requires this delay to be at least 71 microseconds, and a maximum of 100 milliseconds can be used.

The second timing parameter that must be entered is the time interval between samples controlling the sampling frequency. As stated earlier, the minimum interval possible is around 100 microseconds, whilst the maximum allowed by the program is 10 milliseconds to give a minimum sampling frequency of 100 hertz.

Subsequently control is passed to the subroutine H$MPL during which time the sampling process actually takes place. On return

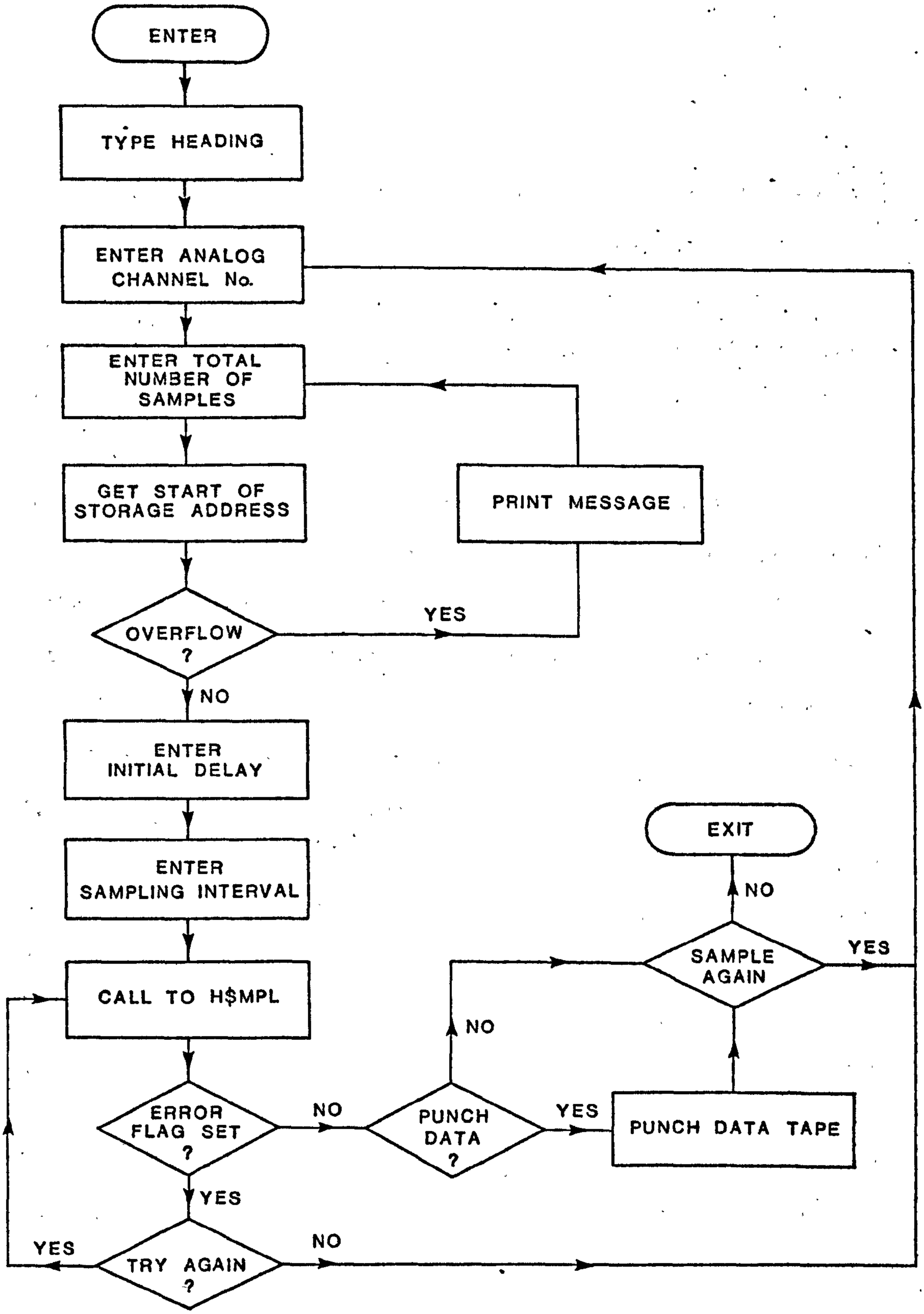


FIGURE 9.1 - H$SP1 ALGORITHM

from the subroutine a check is made to see if the error flag has been set owing to any failure to perform the sample properly. In the case of a sampling failure the operator is given the choice of making a second attempt to perform the same operation or returning to the start of the routine to redefine the sampling operation required.

After the conclusion of a successful sample the array of data can be punched onto paper tape if required. A repeat of the whole sampling process can then take place or alternatively a return is made to the calling program to allow the use of the sampled data by other routines such as digital filters and data analysis programs.

The algorithm of the subroutine H$MPL is shown in Figure 9.2. It performs a check that the analog input channel specified is valid (i.e. between 1 and 32) and converts this number into the computer internal addressing system. The initial delay and sampling interval are converted from their real time values into the actual number of executions required for the software delay loops that generate these periods. The duration of a single execution of each loop is 1.92 microseconds and so the periods are internally set to the nearest whole number of loops representing them. Any error in channel number or time period definition that is detected causes the error flag to be set and a return to the H$SP1 routine without any sampling having taken place. If, after sampling has taken place, the error flag has been set due to an actual failure in the sampling process itself, a message to this effect is printed before the end of the subroutine.

The actual sampling process is carried out by a subroutine

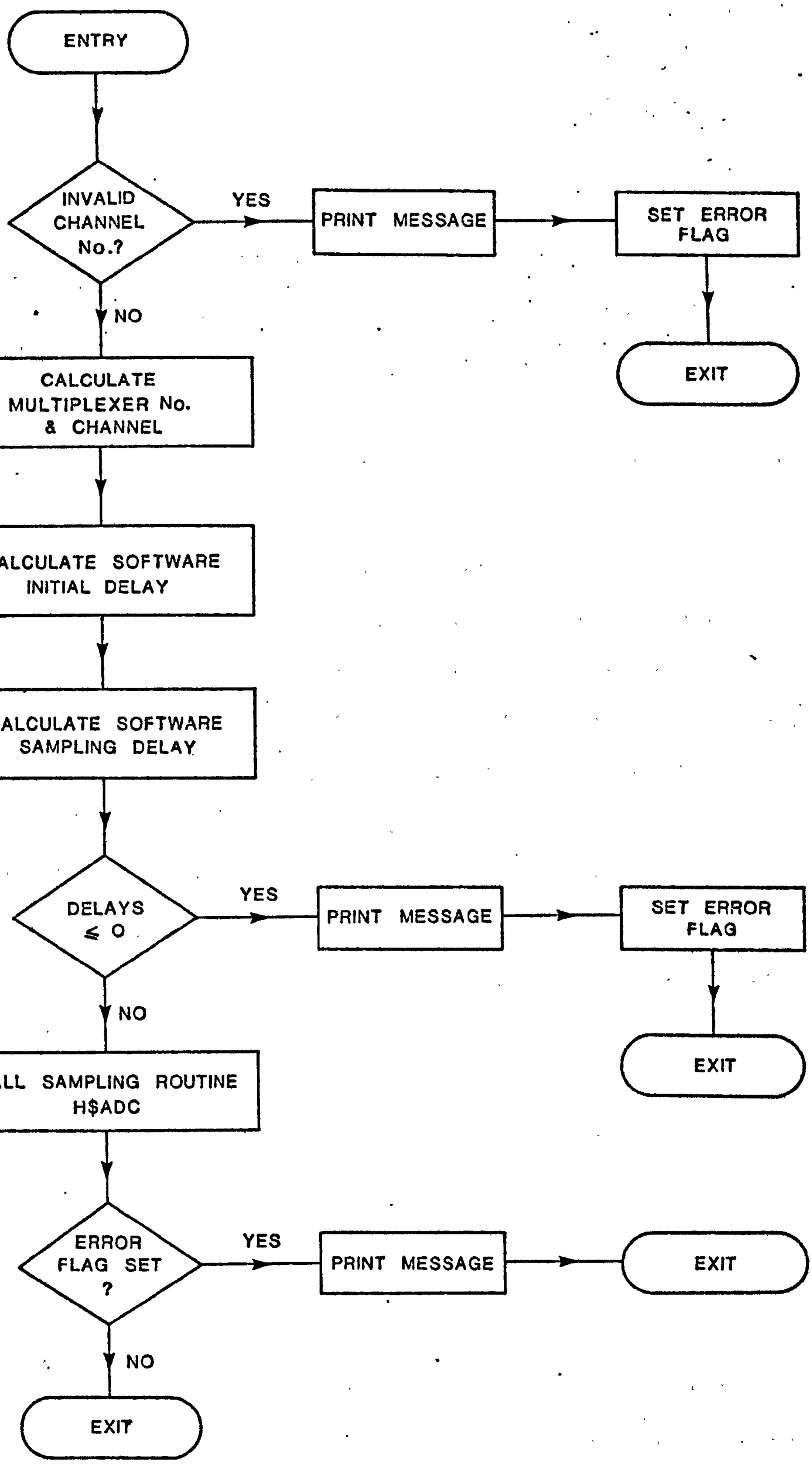


FIGURE 9.2 - H$MPL ALGORITHM

called H$ADC (see Figure 9.3). Unlike the other two programs of the package which are written in Fortran, this routine is in the processor assembler code. This was necessary because of the use of interrupt driven sub-sections, direct input/output handling and the need for a high degree of programming optimisation to minimise software execution.time overheads during the duration of the sampling operation.

The first part of the program is concerned with settling up the various software and hardware parameters ready for the actual sampling. On completion the program enters an endless loop to await the occurrence of an external 'start sample' interrupt. From then on the operation of the program can best be illustrated by examining the timing diagram of Figure 9.4. Each actual sample is preceeded by a period taken up by some program execution time (used to output the multiplexer select command)and the delay due to multiplexer set-up time. These delays must be included in the initial delay period before the first sample and are deducted from the value entered by the operator. After the start sample interrupt a software delay loop operates for the remaining amount of delay.

The actual voltage sample is performed virtually instantaneously by the start conversion command, although it then takes another 40 microseconds until this voltage has been converted to its digital equivalent. After the start of conversion, a software ready flag is reset to indicate that the conversion is incomplete. The program then enters the sampling interval delay loop. At some time during the operation of this loop, a 'conversion complete' interrupt should

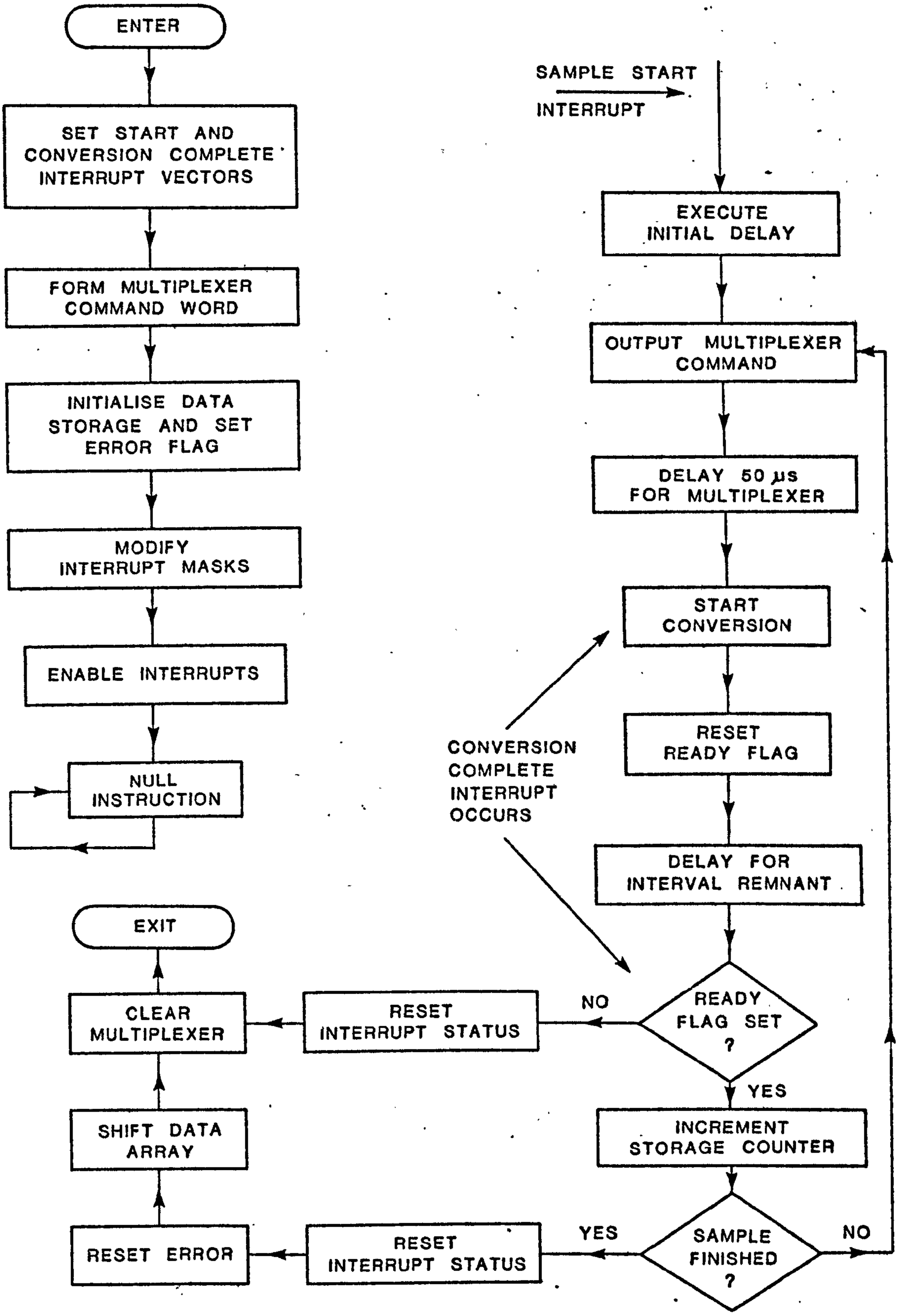


FIGURE 9.3 - H$ADC ALGORITHM

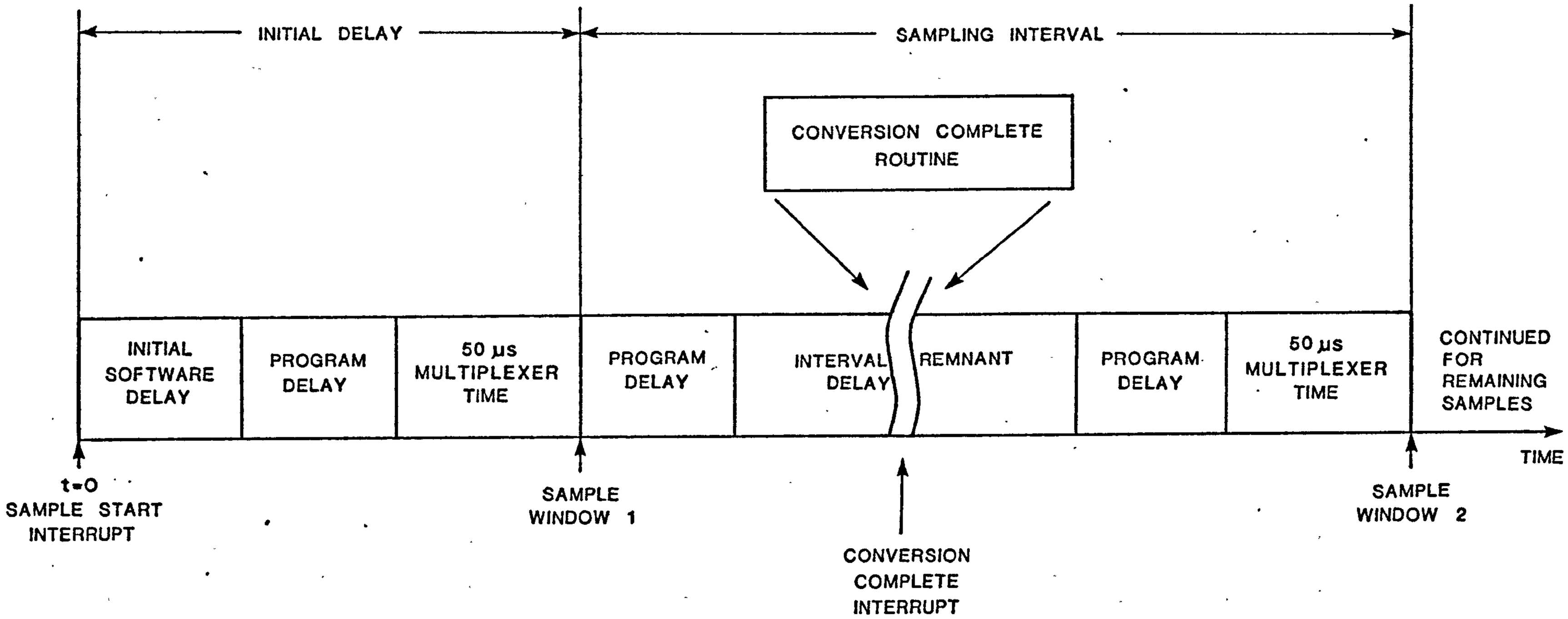


FIGURE 9.4 - SAMPLE TIMING

occur, causing a vector to the conversion complete service routine (Figure 9.5). The first action of the routine is to test the analog to digital converter hardware flag, to check that the conversion is in fact complete as the computer system at times responded to spurious interrupts due to electrical interference. Such an interrupt is ignored, and the delay count adjusted to allow for the time taken in vectoring to the routine and checking the validity of the interrupt.

If the conversion is in fact complete, the software ready flag is set to reflect this and the digital word transferred from the converter to one of the processor registers. The converter itself is a 12 bit device with a range of ± 5.1175 volts and the 12 bit result of the conversion occupies the low-order 12 bits of the 16 bit register. The high-order 4 bits are masked out (set to zero) and the resulting word is transferred to the next location in the assigned data storage area. This completes the routine and execution returns to the main software delay loop.

On completion of the software delay, the program checks to see if the ready flag has been set. If set, the last sample is now safely in memory and all that remains is to increment the storage address counter and to check to see if the required number of samples has been made before returning to the start of the sampling loop. The actual duration of the software delay loop between samples will obviously be less than the sampling interval specified, as the time taken to execute the other instructions including the conversion complete routine has been deducted by the program H$MPL.

If the program at any time finds the ready flag has not been

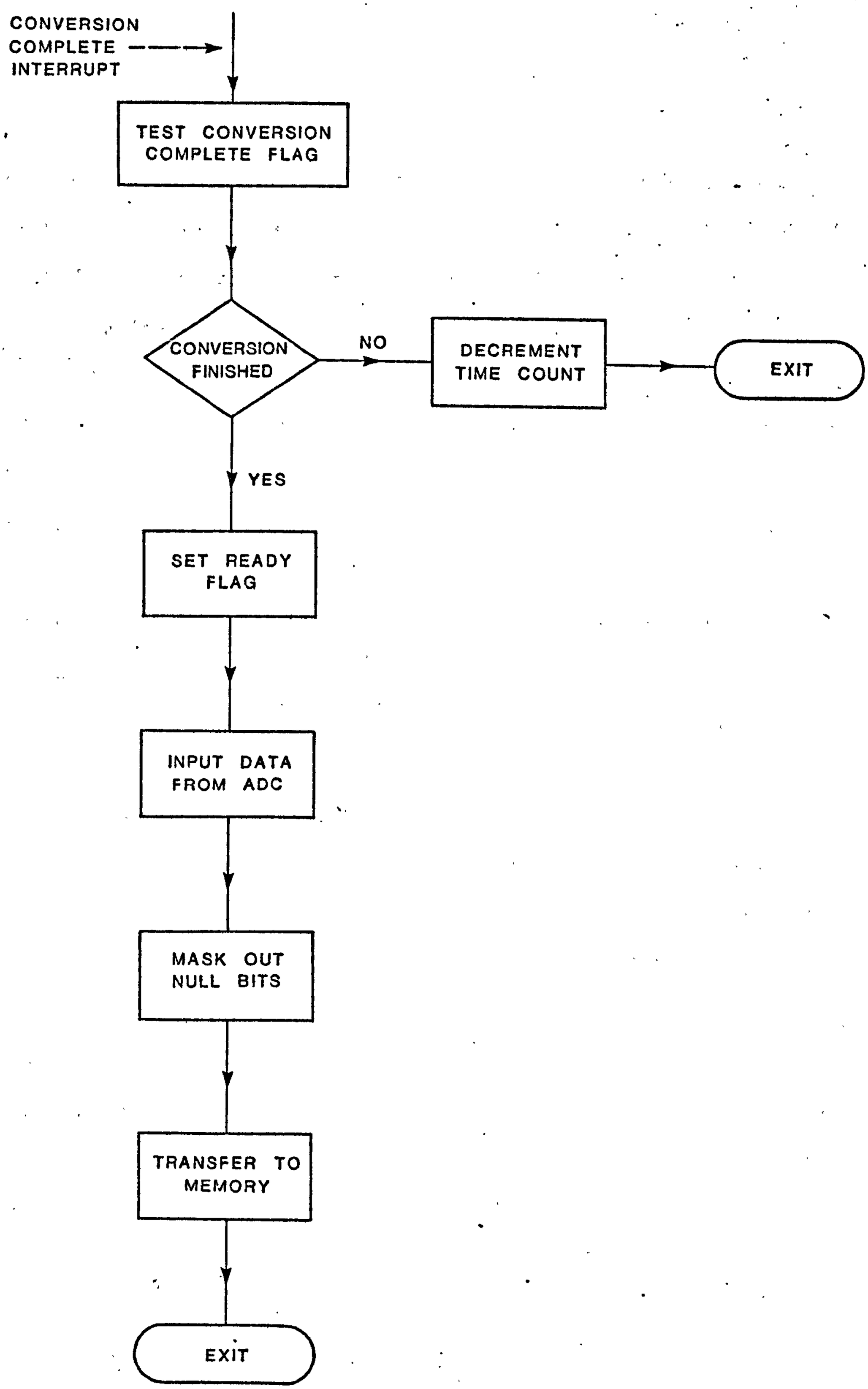


FIGURE 9.5 - H$ADC CONVERSION COMPLETE ALGORITHM

set on completion of the delay, this means that the sampling has failed (probabily due to the request for too small a sampling interval). The sampling attempt is then immediately aborted and after resetting the hardware, the routine exits.

The hardware is also reset following the final sample. In addition the error flag is cleared. Each word in the memory data buffer is then shifted so that the conversion result is in the high order 12 bits of the word. As a result the data obtained has a range of ± 32,752 representing the voltage range of ± 5.1175 volts. It is up to subsequent routines to convert the data into real voltages should this be required, although as in most diagnostic work it is the relationship of one data point to others that is important, this is often unnecessary.

The overall package provides an easily used mechanism for high frequency data logging of a single parameter. Multiple logging could also be handled in a similar way by scanning the relevant channels although this would lower the maximum frequency and any analysis routines would have to take into account the time skew resulting. The package may be called as a standard FORTRAN subroutine and set to return with the sample data stored as an array in COMMON or GLOBAL.

In order to preserve the integrity of any diagnosis technique using this sampling method it is important that the timing of all operations during the sample is precise and repetitive. Because of this the technique is not well suited to a multiprogramming type environment where the routine may fail to obtain a share of execution time at precisely the right instant. This will have an important

bearing on the architecture and automation strategy of any system designed for this form of testing.

9.3 Data Analysis

It is not proposed to say a great deal about the processing and analysis of the data once it has been logged as this work is essentially mathematical in nature and can be regarded as somewhat separate from the testing of engines and its automation. It does not effect the actual system hardware or test software. The following section is therefore only intended to give a broad outline of the way in which data from the high frequency sampling routine can be used to indicate engine malfunctions and other work (43, 45) should be referred to for a more detailed look at the subject.

9.3.1 Digital Filtering

In many cases, prior to evaluating the information contained within a signal, it may be necessary to perform filtering operations to remove frequency components outside the bandwidth of interest. In many cases this can be done simply by processing the data array through the mathematical equivalent of an analog filter. For instance a first order low pass filter can be implemented as follows:-

$$B_r = \alpha A_r + (1 - \alpha) B_{r-1} \qquad (9.1)$$

for r = 1 → n

where A is the unfiltered array

B is the filtered array

Both arrays consist of n samples numbered 1 to n with time.

There are certain cases in which the above digital process does not match its analog equivalent. One factor to be taken into consideration is the fact that when calculating B_1, the term B_{r-1} is non-existent. The way round this is to make it a special case and use A_1 instead. This however will simulate a filter which was only switched on at the start of the sample and hence there are transient effects to be allowed for. The normal way of getting round this is to use a sampling period duration which is more than adequate to allow the filter to settle and to ignore the early part of the results.

Similar to the technique above, other digital filters can be constructed to simulate their analog elements.

9.3.2. Time Domain Analysis.

Analysis of a signal in the time domain can often show up information about the process from which that signal was derived. One such technique is to obtain the frequency spectrum of the signal by using Fourier analysis. The use of Fast Fourier Transform mathematics makes this a fairly simple process that can be performed as part of the data processing in the automation system. The

presence of faults will change the relative strengths of the various frequency components in the signal. An example of this can be seen in Figure 9.6. The first trace is the normal spectrum of the signal obtained from an accelerometer mounted on top of the engine to measure vibration. The second trace shows how the spectrum changes with the failure to fire of one of the engine cylinders.

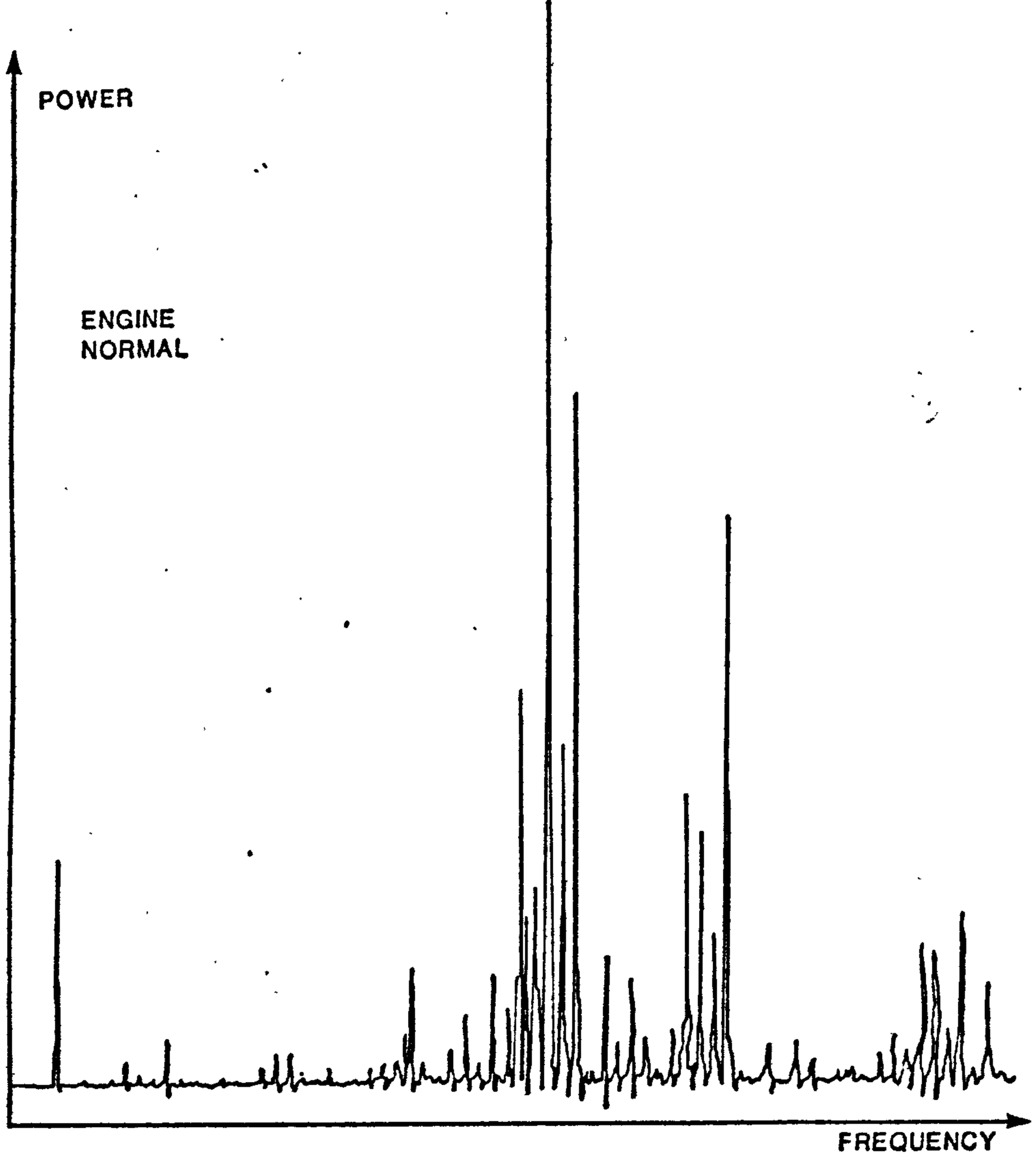
9.3.3 Feature Extraction

The use of feature extraction as a diagnostic method involves use of a particular element or property of a complex signal to give an indication of the information about a particular process element which is contained within the signal. Because feature extraction tends to be specific to each particular problem it is difficult to describe it in general terms and instead it is illustrated here by an example.

Figure 9.7. shows the signal representing the injector pipe line pressure on a diesel engine. The actual plot is the data obtained from a sampling process using the previously described package. The second trace (Figure 9.8) is a repeat, except that slight leakage has been allowed to occur by slackening off the injector itself by a very small amount. The leakage was too small to be visible.

As a result of this leakage, we should look for some change in the pressure waveform. The most detectable feature is lack of residual pressure during the oscillations. Thus we can use the size of the area under the lower curve boundary (marked as A in Figure 9.9) as our discriminatory feature. The way to extract this in-

VIBRATION MEASURED BY ACCELEROMETER ON ROCKER COVER



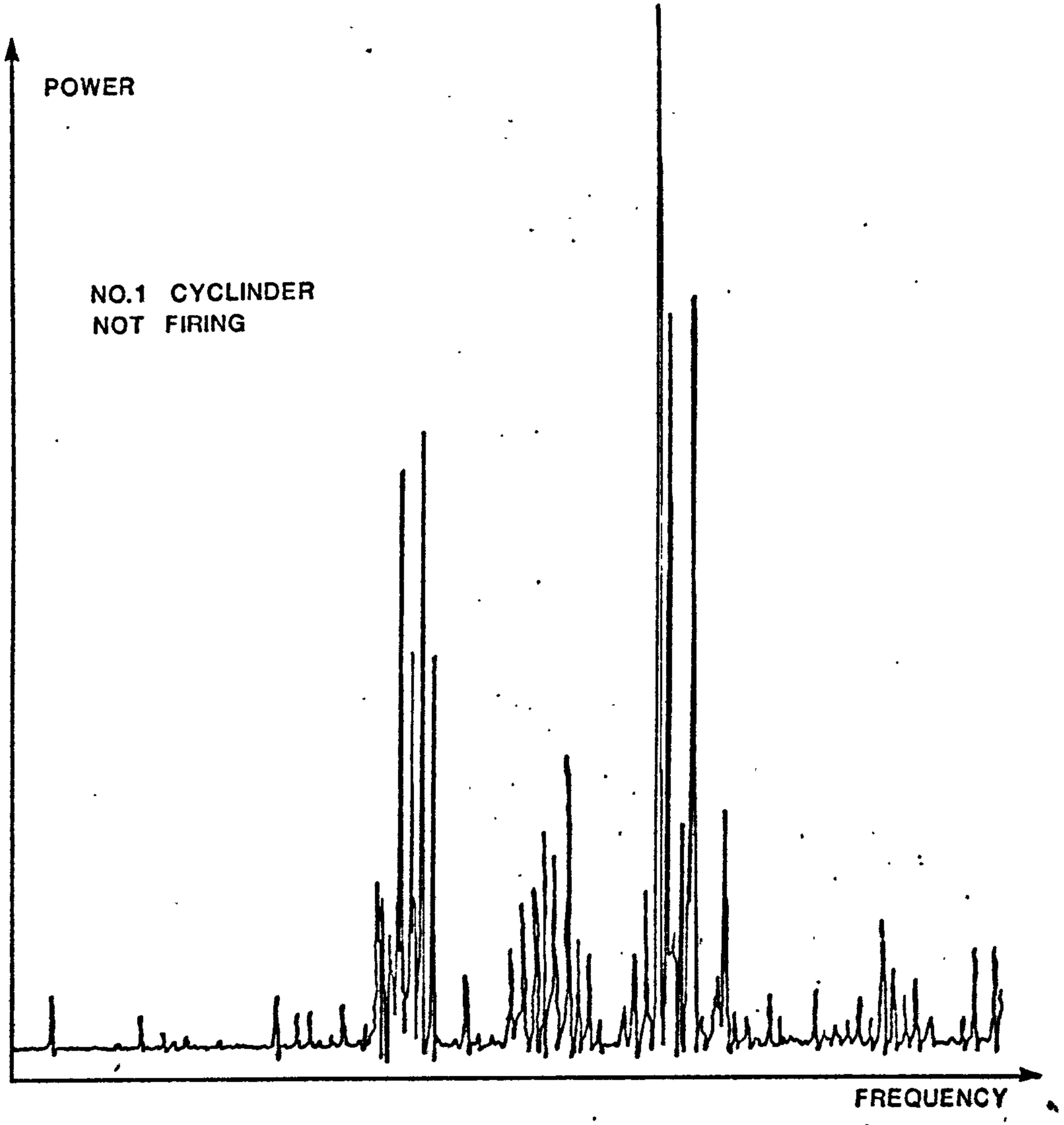


FIGURE 9.6 - FOURIER ANALYSIS OF ENGINE VIBRATION

FIGURE 9.7 - INJECTOR PIPE PRESSURE

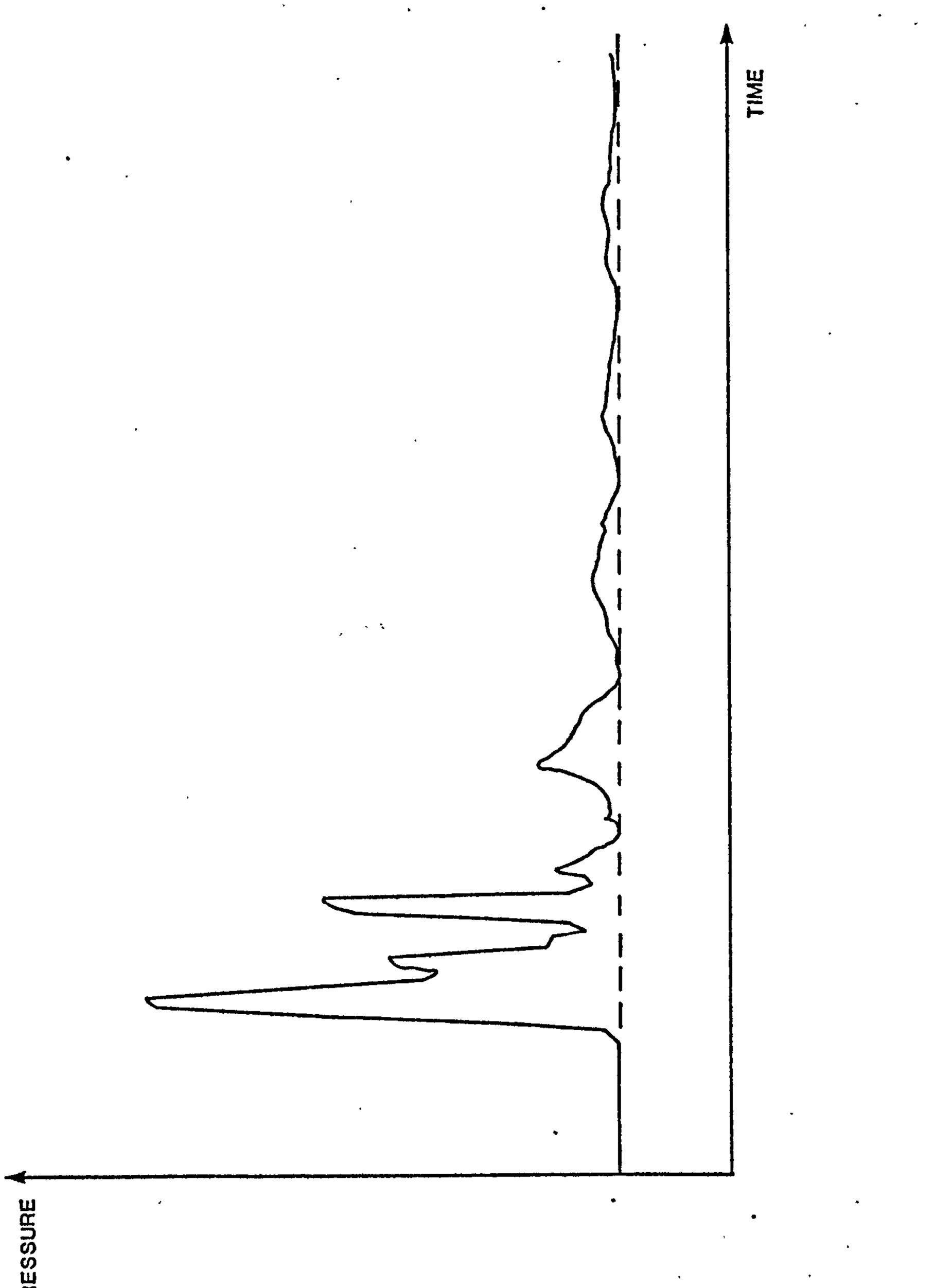


FIGURE 9.8 - INJECTOR PIPE PRESSURE WAVEFORM (WITH SLIGHT LEAKAGE)

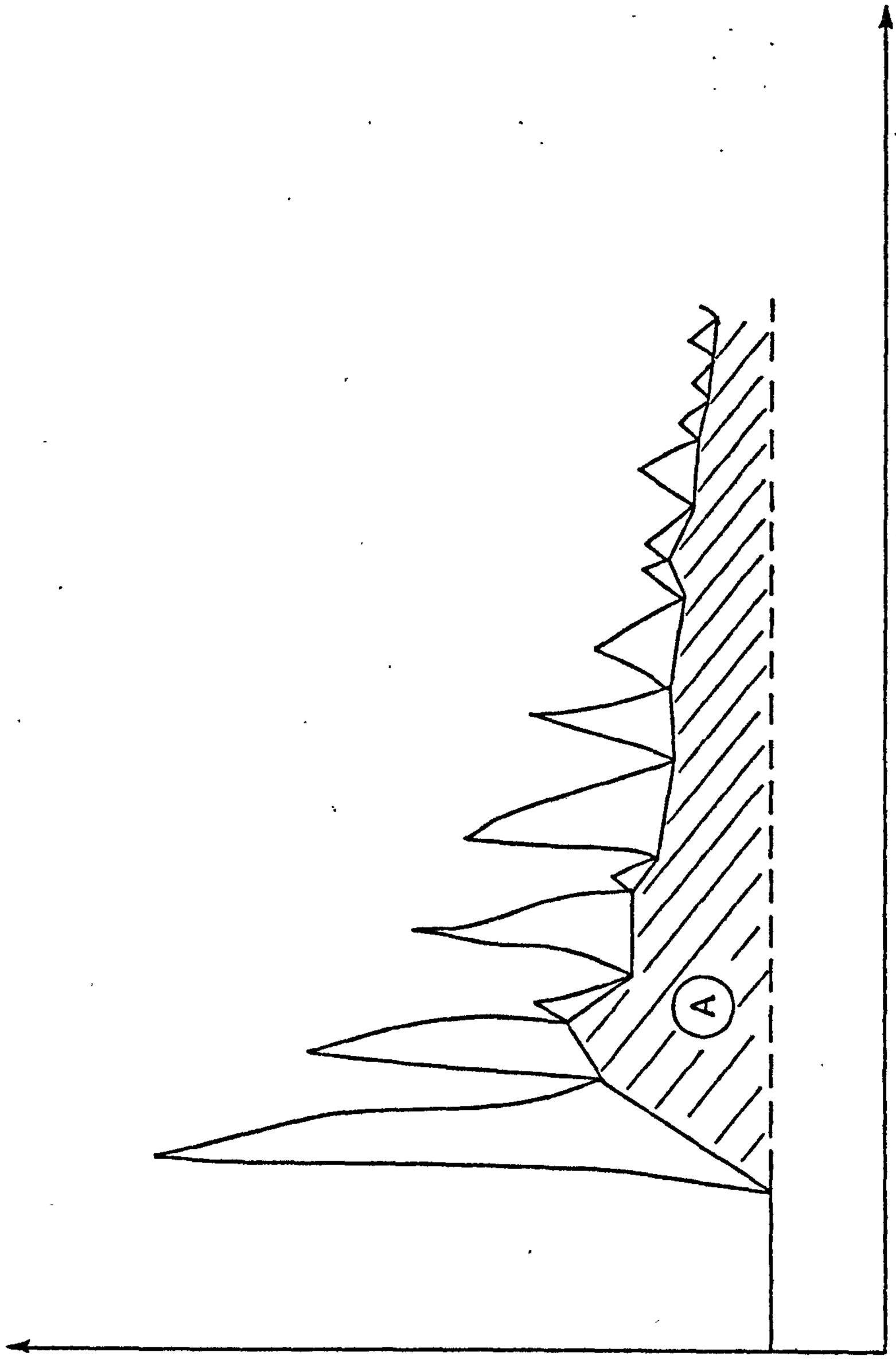


FIGURE 9.9 - INJECTOR LEAKAGE DISCRIMINANT FEATURE

formation is to scan the array to provide the new feature variable A such that:-

$$A = \sum_{r=a}^{r=b} M_r \qquad (9.1)$$

where $M_r = X_r$ for $X_{r-1} > X_r < X_{r+1}$

and $M_r = 0$ for all other cases.

In this case X_r is a data item from our array of sampled data and a and b are the start and end points respectively of the section of the array pertinent to the operation of the process under consideration. It then remains to find from experimentation some minimum value of A that represents adequate sealing of the injector system.

9.4 Summary

In the case of engine test systems which require specific fault diagnostic abilities rather than the provision of general performance data, somewhat different system capabilities are required. Two basic approaches to selective diagnosis can be used. The first involves the use of special purposes sensors or actuators (such as a system to disable one cylinder at a time for the power contribution test). An alternative approach is to obtain a detailed picture of the way a particular engine parameter behaves at any given instant, rather than using a value which represents an average of the process

over one or more engine cycles.

The chapter shows the way in which the high frequency data logging technique, necessary for the latter approach, can be implemented and describes some of the methods which can then be used in the results analysis stage. The use of high speed logging techniques requires either a special dedicated set of hardware or the use of a single task software processing system to provide the high precision timing vital to the operation of the system. This consideration will obviously have important ramifications in the design philosophy of any engine test automation scheme which includes such diagnostic capabilities.

Chapter 10

MICROPROCESSOR BASED SYSTEM

10.1 Introduction

Recent years have seen a revolution in the field of digital electronics. It is true that the complexity of the integrated circuit has been increasing for many years. For a long time though, this only meant that we were getting a larger number of identical, simple circuits per chip. There was little evidence of any use of the advances in the technology of integrated circuits to increase the complexity of the pre-built functions available from the manufacturers. Thus to produce any involved piece of digital processing equipment still required a high level of design expertise and this resulted in high cost products. However a great change occurred when it became possible to build enough gates into a single chip of silicon to provice a complete central processing unit within an integrated circuit. This device, known as the microprocessor, has meant that complex functions can be implemented for very little cost. The microprocessor, together with a range of support circuits such as memories, input/output controllers, etc. can be used to build a system which requires little expertise beyond a basic understanding of digital logic to put together the hardware. As a result these devices have already found a wide degree of usage in industry. Often they have been used to replace hardwired circuits of considerably less complexity than themselves but the reduction in cost resulting from the use of standardised hardware combined with increased flexibility due to the use of software rather than hardware function control still makes the microprocessor use attractive.

Before considering the actual usage of microprocessors in the

future of engine test automation, we should first consider the requirements that we will set out on the basis of our experience to date.

10.2 Future Automation Requirements

In specifying the requirements for our next generation of systems, there are two seperate areas which should be considered; these being the drawbacks of existing systems and the new testing requirements that future systems will have to cope with.

10.2.1 - Historical Experience

Several examples of existing test facilities are described in chapters 2 and 3. Considering these facilities together with the many other examples of 1st generation fully automated systems, there is one almost universal trend in the basic automation philosophy used. The basic automation element (normally a minicomputer) is used to handle a number of independent asynchronous real-time functions. In most cases at the highest level this involves the supervision of several test beds possibly running different tests, on different engines, all at different times. To make matters worse there is a further complexity caused by the fact that there are several separate tasks being carried out for each engine under test such as data logging and alarm monitoring.

The spreading of the computer processing power among so many different tasks was done in an attempt to make 100% use of the system capability. This in turn was necessary because of the comparatively

high cost of minicomputer systems. It would not be economically justifiable to use one minicomputer to automate a single test bed and so several would be connected to the computer instead.

This multitude of tasks invariably meant one thing; the software for the system would have to be based around a real-time operating system to provide the multi-programming environment. The technique of making a single processor behave in such a way that several users appear to have access to their own individual systems (virtual machines) is well established in other fields and was not thought of as being likely to prove problematical in this application.

Unfortunately experience has not proved this to be the case. One common fault found in many of these first generation systems was that even after the individual applications programs had been written, and thoroughly de-bugged, considerable problems would be experienced in trying to get all the different routines to behave properly under the control of the operating system without software 'crashes' caused by improper interaction between different routines. Such problems are very difficult to isolate and solve, owing to the fact that they only occur when the system is operating in its normal method and the usual methods of tracing software bugs using step by step execution of programs under constant monitoring of the programmer will not show up the causes.

The basic reason for the occurrence of such problems in an engine test environment when the technique of real-time multiprogramming is already well proven in other industrial process control applications lies in the area of time response requirements. In many of the

applications for which the technique was originally developed, such as the petro-chemical industry, typical time scales in relation to process changes are comparatively slow and system response-to-stimulae times measured in seconds acceptable. On the other hand when we consider an engine under test it should be remembered that in certain conditions it is possible for an engine to 'run away' to destruction, due to overspeed, in under a second. As a result, such facets of the system as alarm condition monitoring have to be run at comparativ-ely high frequency. Thus if we take the case of a facility with ten test beds under a single minicomputer with speed and torque alarm checks being sequenced at half second intervals, then the alarm moni-toring requirements alone demand 20 different program execution tasks per second. Add to this the other parts of the computers task structure, such as set-point sequencing, data logging, data processing and test data output, and it becomes apparent that we are talking about an environment where the operating system is being required to perform its typical operations of swapping programs and data in and out of bulk storage, scanning inputs and outputs, etc., at very high frequency.

On the face of it there is still no reason why the computer should not be able to handle such work. The problem arises in de-tecting the cause of faults due to the difficulty in following exactly what is going on in the computer at any given instant. In one case, a small software error, the overlapping assignment of a particular area of memory as data storage to two different programs simultaneously, took many months of expert software personnel time to solve. The problem was not the correction of the error, a simple matter in itself,

but in isolating it in the first place.

The upshot of all this is that what at first seems the most economical solution in terms of automation philosophy, has a hidden cost factor, namely the increased cost due to the increased amount of expert programming personnel input required and the delay in development and commissioning of the system. Furthermore the complexity of the total software package is such that any modifications or improvements required during its lifetime will only be effected with a considerable amount of difficulty.

10.2.2. - New Techniques

The first generation systems described above were based around the use of what were basically steady-state testing methods. In the earlier chapters we have considered the sort of testing techniques which should be included in the capabilities of the next generation of systems. The most obvious facet of the work is the use of dynamic testing techniques and this will have a critical influence on the choice of automation philosophy employed. The prime difficulty arises, as might be expected in a dynamic environment, from the point of view of the timing requirements of the system. If the time response in steady-state systems has given rise to problems, the situation with regards to dynamic testing is even more critical. The operation of dynamic tests not only requires high frequency response in terms of control and data logging but also high precision of scheduling.

In the steady-state testing environment we have talked of

running alarm programs at half second intervals, whereas for dynamic tests such as emission cycles we will be updating set-points to form ramps at perhaps 10 millisecond intervals. Additionally the slight delay of, say, a data logging scan with any engine at steady-state operating conditions does not matter, but for a dynamic situation the precision scheduling and execution of the scan is vital as even a very small delay may invalidate the results obtained. Thus if the time response behaviour was thought to give rises to problems for steady-state testing, it can be seen that for dynamic testing, it effectively precludes the use of a system philosophy involving the control of any appreciable number of test beds by the same processor operating in a real-time multi-programming mode.

10.3 Solution Approaches

There are several different approaches that can, and in some cases, have been used to get round some or all of the problems described in the previous section. Their nature and relative advantages and disadvantages are discussed below.

10.3.1 - Core Only Memory

This approach attempts to simplify the multiprogramming problem by avoiding the need to share the available core memory between several different programs. Instead of holding the majority of the software in bulk memory (usually a magnetic disc or drum) and performing the necessary roll-in and roll-out operations between bulk and core memory, a very large core memory is provided. The soft-

ware is written as though single test bed operation was being used and the resulting package is duplicated in memory to provide one copy per test bed in the multiple bed environment. The work of the operating system is therefore simplified and the risk of software core requirement conflicts avoided.

There are several disadvantages to such a system. The technique does not do much to simplify the actual scheduling of routines, even if it does eliminate any delays resulting from a required program not being present in the core memory. Also the hardware structure of most minicomputer systemslimits the total amount of core memory that can be added on, thus requiring either a fairly simple and limited software package and/or a small number of test beds under the one computer. The duplication of identical software is somewhat wasteful, especially in view of the high cost of main memory compared to bulk storage memory.

10.3.2 - Table Configured Standard Packages

In an attempt to reduce the costs and problems arising from the development of the software for use in engine test automation schemes, some computer system manufacturers have advanced the concept of using software packages based on a standardised package which can be tailored to a particular application by the generation of configuration data tables. Two examples of these packages are G.E.C.'s CONRAD (39) and General Automation's TOPICS (22). Both these systems have been used in engine test automation and certainly prove successful in their basic objective of simplifying programming require-

ments. The use of pre-written and tested standard programs avoids the risk of programming errors and the table based organisation of the software makes it possible for package generation and modification to be performed by personnel with comparatively little or no programming expertise.

The disadvantage of such a system is that it is virtually impossible for any standard package to meet all the detailed requirements of a particular automation scheme exactly and it may well be that the user of such a package will have to accept a compromise approach that is not exactly what he wants. Furthermore the suitability of such packages for dynamic testing is distinctly questionable. Inevitably any attempt at producing generally applicable programs will result in a certain amount of inefficiency when compared to a routine specifically written for the task in hand and so the application of such packages will tend to be limited to environments where the time response required of the system is quite slow, such as durability testing or steady-state power curve work. Nevertheless for these applications, the concept of the table configured standard package will provide a cheap, easy to implement approach, but for more demanding test systems a different philosophy will be required.

10.3.3 - Distributed Processing Networks

When reviewing the problems posed in section 10.2, it was felt that the first stage in drawing up plans for future engine test automation systems should be the isolation of the real problem areas. It became apparent that the prime difficulty does not lie in the

actual application programs, but in getting them to run properly in a real-time miltiprogrammed fashion. The easiest way to get around any such software problems is to let the hardware do the work. The ultimate simplification of the problem would be to run every different part of the software in its own individual processor thus completely obviating the need for any form of time sharing. This concept of a network of distributed processing elements would have the effect of minimising software complexity but would also require a considerable amount of hardware. Traditionally, when the cheapest form of digital processor available was the minicomputer, such an approach would have been far too expensive to consider. The advent of the microprocessor which can provide us with very low cost simple processing systems however makes such an approach far more attractive. The remainder of this chapter describes the way this type of approach could be used for engine test automation.

10.4 System Task Analysis

The first step in setting about the choosing of the automation philosophy and structure to be used is to draw up a comprehensive list of the tasks that must be implemented in the operation of the system. From this we can analyse what the requirements are for each task in terms of their own behaviour and their interaction with the other tasks.

The broad range of the tasks is shown as a spectrum in Figure 10.1, ranging from high level work such as management reporting on the left to the basic interfacing with the test bed itself on the

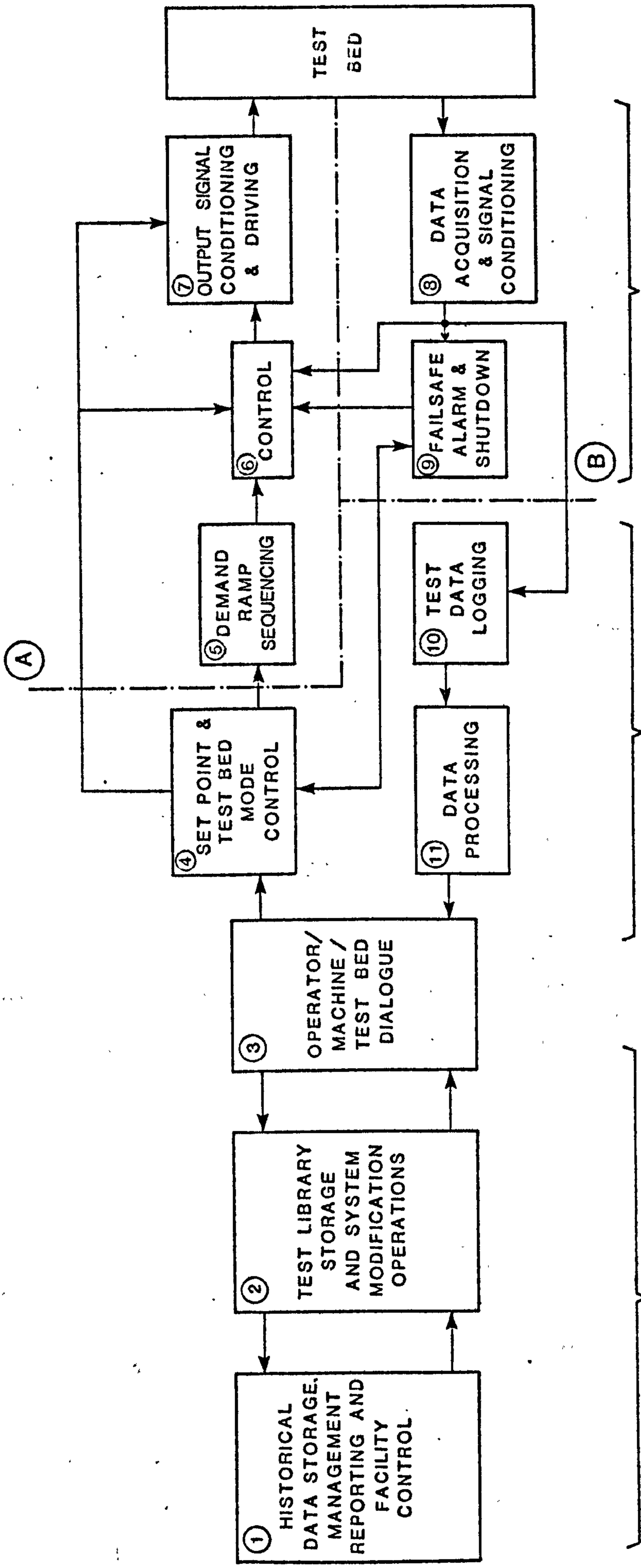


FIGURE 10.1 - SPECTRUM OF ENGINE TEST AUTOMATION TASKS

right. Each of the various sections is described in turn below.

Section 1 - This section includes the overall supervisory work of the whole test facility including such tasks as historical data storage for management reporting and trend analysis, and perhaps the control of an associated handling system. Obviously not all facilities will include this level of task, but such work is more likely to be found in a production test environment where this section might also include selection of the test procedures for a given engine.

Section 2 - In systems which do not include the type of work of the section above, this forms the high level of the overall workload. The operations here are storage and handling of the library of programs that make up the facility test system. In addition most facilities will need to incorporate the ability to modify and update their own programming to take account of changing requirements and the mechanism for doing this (compilers, core load builders, etc.) are included at this level.

Section 3 - The operator must be able to communicate with the automation system and the test bed. His communication with the test bed may only be indirectly by way of the automation system but in most cases direct communication in the form of manual control and direct instrumentation readouts will be available. The normal man/machine dialogue would include test sequence requests, system

behavioural alteration (such as a change of alarm limits), operations information (such as notification of an alarm condition) and all data output of both printed data logs and graphical display.

Section 4 - After the operator (or the computer) has selected the test sequence the system must select the required mode of operation of test stand, take over control of the set points from the local (manual) controls and thereafter output the necessary basic set-point updates to perform the test before finally returning the test bed to local control. This section would also normally include any outer digital control loop activity.

Section 5 - If the system is to have the ability to run dynamic cycles then there is a need to generate ramps by a series of small steps at rapid intervals. This has been shown as distinct from the previous section because of its high frequency regular characteristics rather than the lower frequency work of steady-state set point control.

Section 6 - Almost all forms of testing require some form of closed loop control of one or more test stand parameters. This usually takes the form of three term (proportional, integral and derivative) controllers with sometimes other special control functions.

Section 7 - The final stage is the output of signals to the test bed. This section includes the actual electrical interfacing.

It may also include signal magnitude limiting and even rate of change of signal limits.

Section 8 - To perform virtually any test operation the system must be able to access data from the test bed sensors. Although the frequency at which this logging must occur varies from very high frequencies for diagnostic testing to comparatively low frequency for steady state testing, high frequency logging will nevertheless be a basic requirement to allow operation of some control loops and alarm monitoring. Some of the signals derived from the sensors may have to be subjected to a certain amount of signal conditioning.

Section 9 - The provision of some form of failsafe alarm and shutdown system is vital to the safe operation of an automated test facility. The parameters subjected to monitoring and the specification of limits should be as flexible as possible.

Section 10 - As mentioned previously, not all tests may require high frequency data logging. Some provision must be made to allow proper selection of information for the generation of the required test results.

Section 11 - This final section involves all the processing of test data. It includes the whole spectrum of such operations, from simple conversion to engineering units, through such activities as the calculation of derived results, such as power and specific fuel con-

sumption up to some of the complex mathematical analysis procedures used in fault diagnosis.

Once all our various tasks have been listed, the next stage in deciding the way in which they will be implemented is to group them into categories. If one of our prime objectives is to avoid the use of multiprogramming in a high frequency real-time environment, then the first grouping we can make is into two categories; those tasks which require precise real-time response and those that do not. The latter tasks are designated as supervisory functions in Figure 10.1. Because there is no reason for not using multi-programming, in their case we can group the operations in this category for several test cells together in a single processor. As a result, some of the tasks in section 3 will have to be excluded from this category owing to the fact that whilst operator/machine dialogue does not need to be on a precise real time basis, it should occur at the test cell itself rather than at the group processor location.

Obviously to avoid the need for multiprogramming, the other half of the task spectrum cannot have any processor responsible for more than one test bed and so we come at once to the microprocessor per test bed concept. Even with one processor per cell, it is still difficult to avoid multi-programming as can be seen from the dynamic power curve system of chapter 5. Furthermore it is felt that in future as much of the automation system as possible should be implemented using digital techniques. This is due to the high reliability and low cost combined with advantages such as ease of calibration and

good flexibility, which can arise using digital hardware in place of analog circuits. Thus we must include the functions such as controlling and signal conditioning within our processing task categories.

The result of this is that there has been a further sub-division made in the various types of real time tasks. It is important to be able to make the distinction between tasks that may require a precise scheduling but operate at comparatively low frequencies, or even perhaps irregularly, and the second set of tasks which have a high frequency repetitive nature. These two different categories of processing (designated Level I and Level II respectively) are perhaps best illustrated by taking specific examples.

One of the Level I tasks is the sequencing of set-point changes, other than updates forming part of a dynamic ramp. The actual requirements for this type of operation will depend on what the overall system is doing. In between tests the function may be non-operational (as the test bed is under direct manual control) and during a test the interval between updates may be quite large or even irregular, if, for instance, the change is required as the result of some criterion other than time as would be the case in a test where stabilization of a given engine parameter represents the end of a stage of the test.

On the other hand, a typical Level II task is the control loop operation. In this case the operation is in almost continuous high frequency repetitive operation with only changes in such inputs as the set-point and possibly the various gain settings.

The sequencing of demand ramps has also been included as a

Level II task. This is because although it is intermittent in nature (i.e. it only operates during a dynamic ramp and is inactive during steady state conditions), when operating it is more akin to a Level II type task than those grouped into Level I. During a dynamic ramp, updates may be performed at intervals of a few milliseconds and this is at odds with the comparatively infrequent operation of most of the Level I functions. Thus it is thought that the best way of implementing set point control is as follows; the Level I function is to output the actual set points in the case of steady state operation. To perform a dynamic ramp, it instead outputs a ramp rate demand together with a mode instruction to indicate to the Level II system that this is no longer a direct set-point output. The ramp sequencing algorithm in Level II remains inactive in the direct mode of operation but on receiving the ramp mode instruction will proceed to generate the necessary set point updates, thus leaving the Level I system free of any set-point modification tasks until termination of the ramp.

Even within the Level II tasks it is thought that a further division in processing responsibilities should be made. Thus we have Level II Section A representing the output functions, and Section B incorporating the input functions. This has been done because of the workload involved in both the processing of control algorithms and the requirements of high frequency data logging. The control algorithms can thus be operated at high frequency to maintain the fast control response needed for dynamic testing without prejudicing the data acquisition capabilities.

A final problem arises in the implementation of the high speed data logging needed for some forms of diagnostic testing. However this should not prove too difficult to achieve. The actual data acquisition section is already based on high speed operation and the only real problem is to transfer the large amount of data from a single burst of sampling to the Level I processing system. As the control requirements of such testing are limited, all other Level I tasks including outer digital loop functions can be suspended whilst the operation takes place. An alternative solution if the above approach is not acceptable would be to utilise hardware to perform a direct memory access type transfer thus freeing the Level I processor for other tasks during the actual sampling operation.

10.5 Hardware Structure and Philosophy

The basic approach to the test system automation philosophy is derived from the previous categorization of system tasks. The overall structure is shown in Figure 10.2. With the reduction in cost constraints resulting from the advent of the microprocessor, we can assign a separate processor to each of our task categories.

A more detailed outline of the Level II Section A hardware is shown in Figure 10.3. The basis of the system is a fairly simple microprocessor with most of its programs stored in read only memory (ROM). The actual operation of the software is determined by a configuration table stored in a small area of read/write memory, which is also used for dynamic data storage. The configuration information is received from Level I via Input Port 1. This same port is

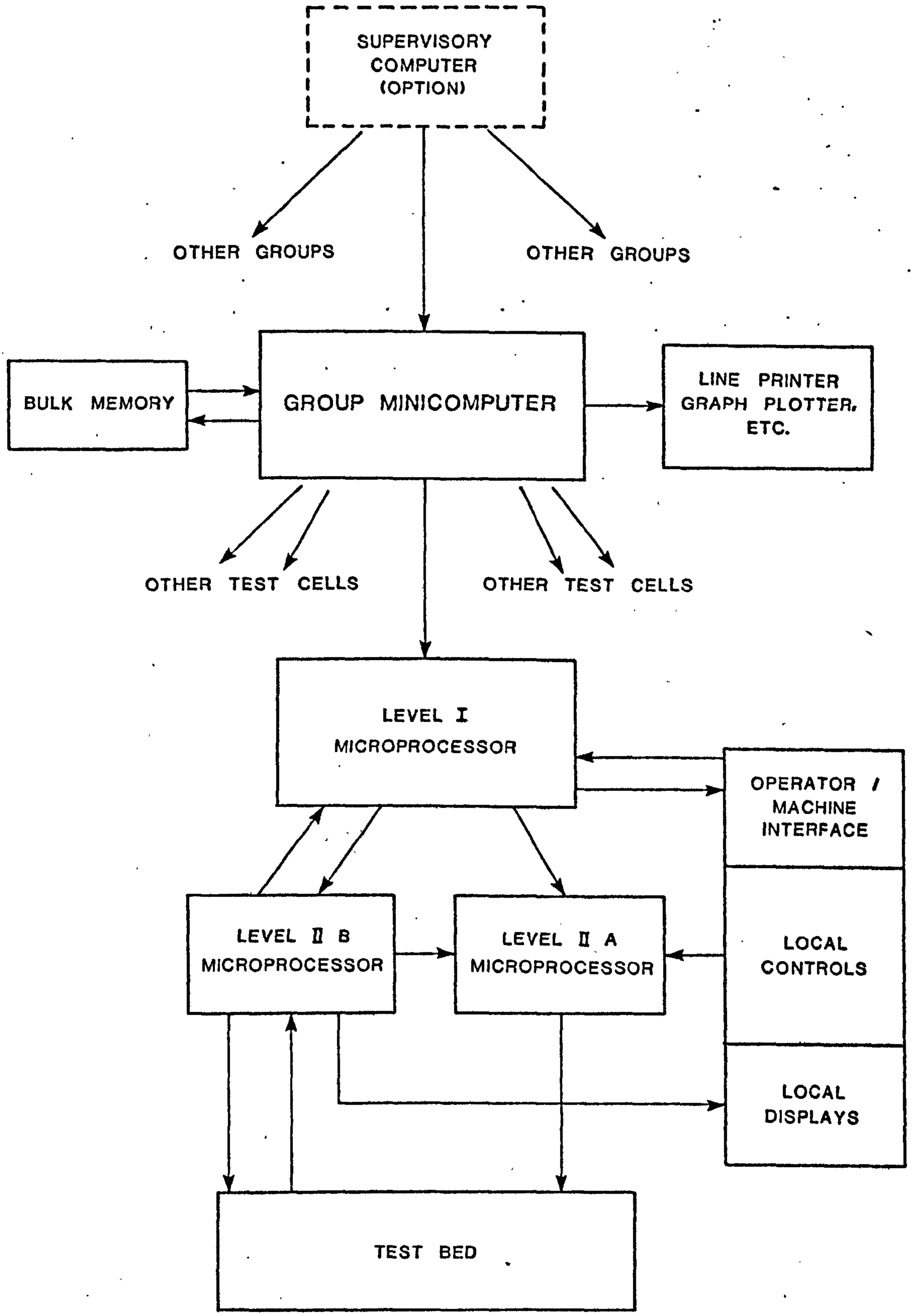


FIGURE 10.2 - OVERALL SYSTEM STRUCTURE

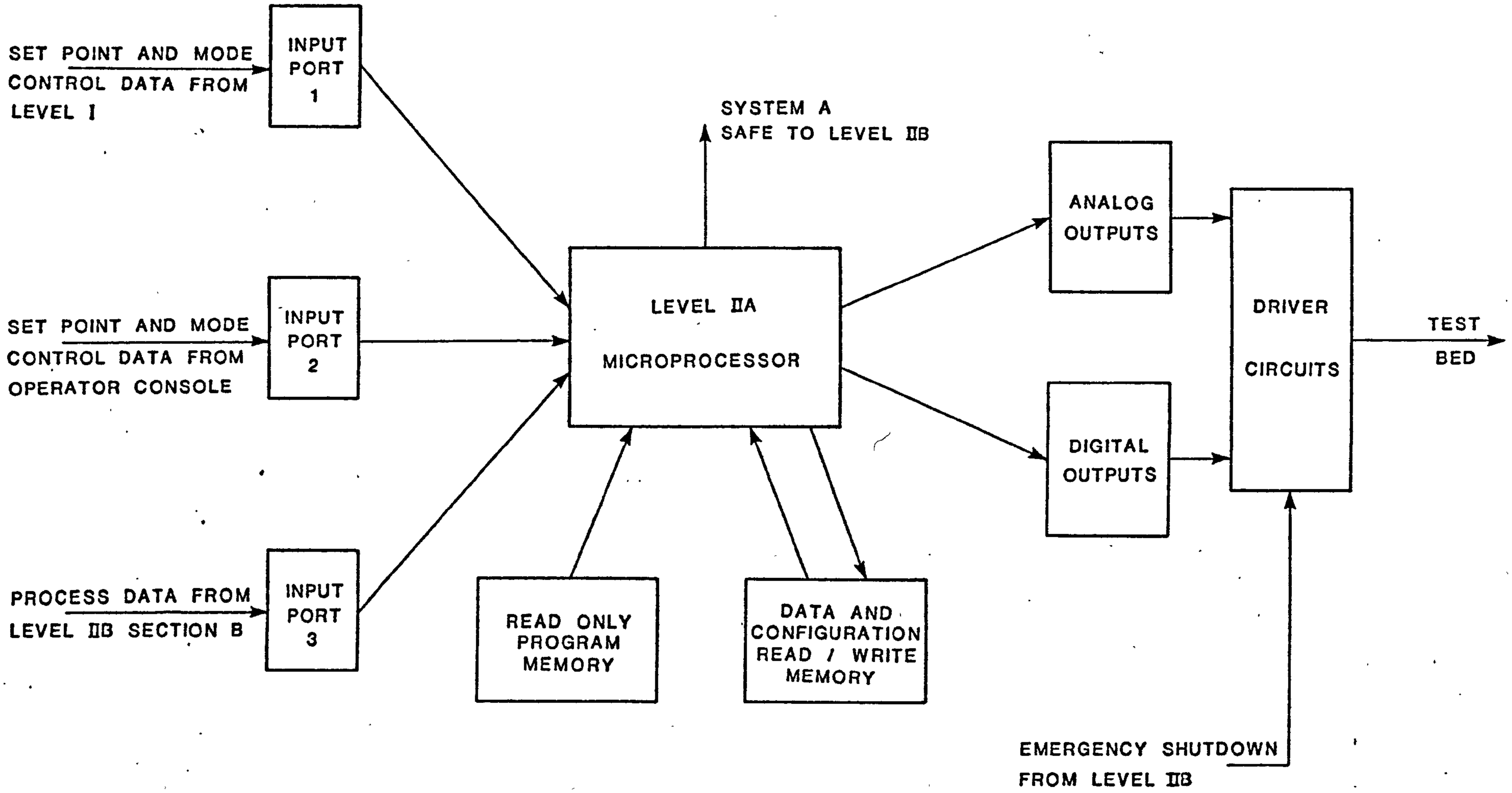


FIGURE 10.3 - LEVEL II SECTION A HARDWARE

also used for the transfer of set point and control mode data when the test bed is under computer control. In the manual mode of operation this information comes instead from the operator console via Input Port 2. Process data for the control loops is received from the input processor through Input Port 3. The control outputs are fed either to digital to analog converters or to direct digital outputs depending on the type of actuator concerned. The electrical driver circuits should be designed so that they can operate in a failsafe mode in the event of an emergency shutdown even if the processor itself has failed.

The input processor of Level II Section B (see Figure 10.4) is very similar in nature. Data from the test cell is fed to the signal interfacing section to provide levels that are electrically compatible with the processing system. Both analog and digital inputs must be catered for and the analog signals converted to digital information using a reasonably fast converter. The microprocessor itself should be able to perform simple digital filtering with its programming again based on a read only fixed memory combined with a configuration table.

The process data together with information on any alarm or shutdown limits which have been violated, is signalled both to the Level I processor (via output Port 1) and to the displays of the operator's console (via Output Port 3). The process data needed for control is output to Level II A from Output Port 2. The control and configuration information is received from Level I through Input Port 1.

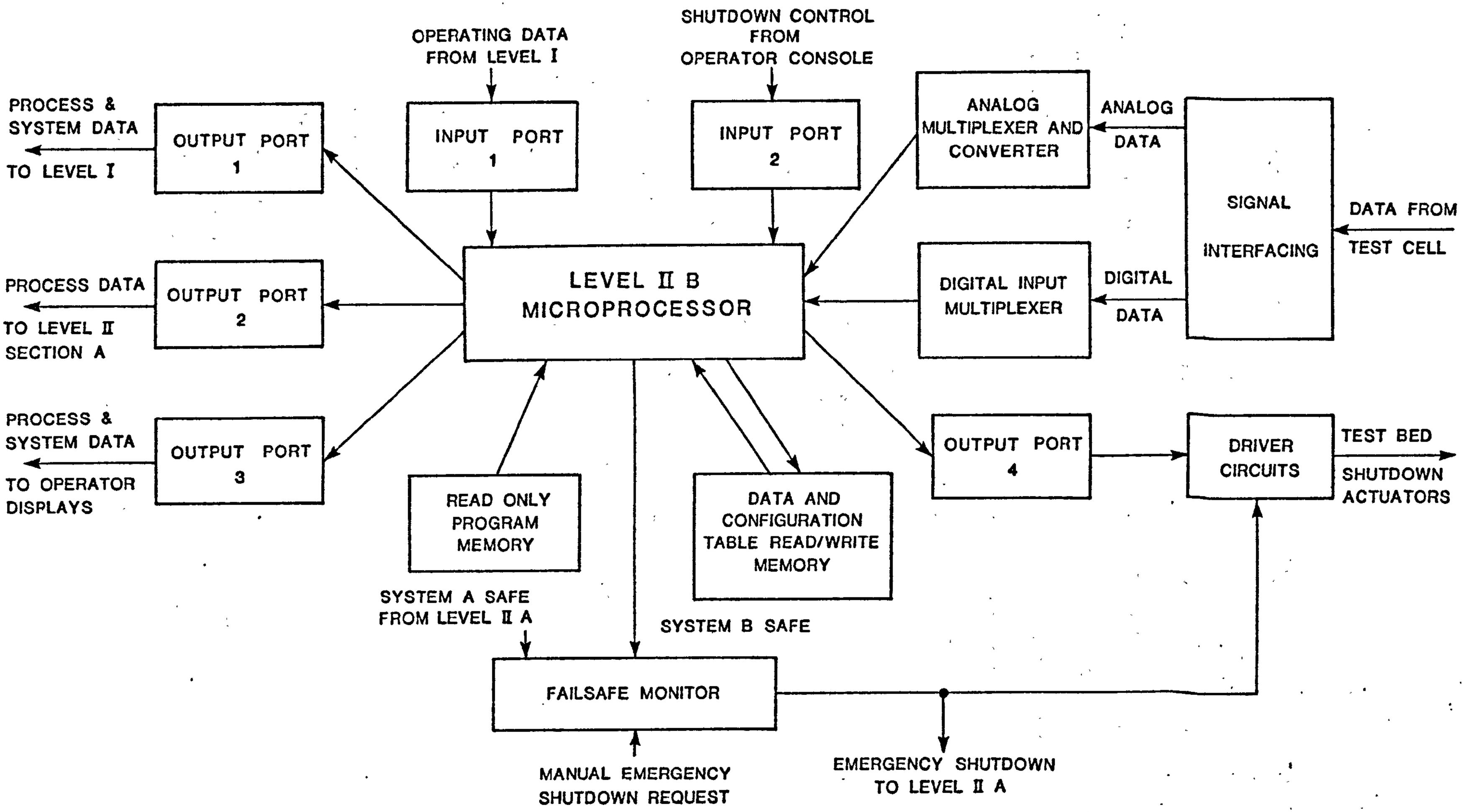


FIGURE 10.4 - LEVEL II SECTION B HARDWARE

This level also has the responsibility for controlling all shutdown operations. Both normal and emergency shutdowns can be either manually initiated from the operator console or as the result of a program function. Normal shutdown requests operate on a program basis using Input Port 2 and Output Port 4 is used to drive the test bed actuators. In addition the emergency shutdown system has a hardwired failsafe function which will perform the necessary operations in the event of a failure in either of the Level II processors. It should be noted that the system can still operate under manual control without Level I processing.

The structure of the Level I processing system can be seen in Figure 10.5. Although still based on a microprocessor, it is much more like a conventional minicomputer in architechture than either of the Level II systems. The memory is almost entirely of the read/write type to allow different test programs to be entered from the group minicomputer library with only a small read-only memory to provide a bootstrap to the operating system. As well as communicating with the Level II processors it can communicate with the operator to allow him to request tests, etc. Another function of its operation could be to output process information to the operator console. This data could be the output from the data processing rather than raw logged data.

The processor must also be able to communicate with the group minicomputer in order to obtain test programs and to transmit test results. In most cases this would probably be done using a serial communications interface to reduce the problems associated with the

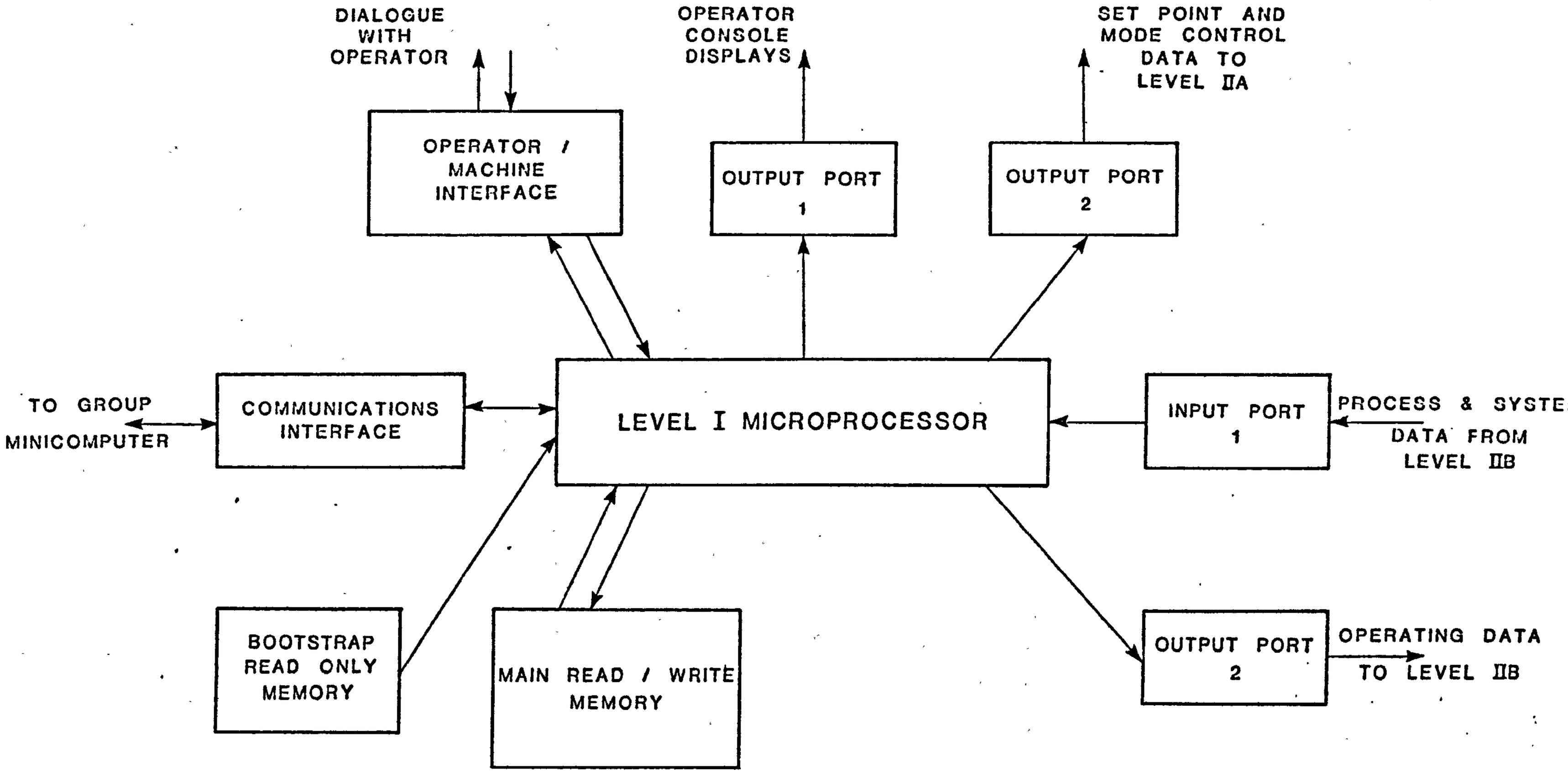


FIGURE 10.5 - LEVEL I HARDWARE

fact that the minicomputer may be at some distance from the test cell itself.

The group minicomputer itself does not require any specialised hardware design. Its main function is to maintain a library of test programs and supply them on demand to the microprocessors at the test cells. On completion of a test, it will accept the results from the test cell system and output them on a line printer or a graph plotter. This avoids the need to have expensive hard copy devices at each test cell, although it does mean that the system is at risk should the minicomputer fail. This is not as big a disadvantage as it might seem, for the group minicomputer is only required by a test cell at the beginning and end of a test so that short duration breaks in service will not stop tests already in progress. In addition the centralisation of the test data printing means that a certain amount of back-up capability could be provided whilst still affording a cheaper alternative to distributed hard-copy printing.

It is apparent that with the minicomputer being connected to a number of test cells, it must have some form of multiprogramming ability. However the time response of the system is not critical and so the software could be based on a standard, well-proven, communications type operating system. There is also no reason why the system could not also be used to perform high level data processing functions which could otherwise require unnecessary complex hardware in the test cell processors.

In a background mode, the system would also be responsible for producing new software packages for the microprocessors at the test

cell to use. Alternatively, in a large facility with many groups of cells in turn reporting back to a supervisory computer, this could be included in the supervisory computer's tasks together with its work in coordinating the whole facility.

Although the number of test cells which can be connected to one group minicomputer will depend on the actual purpose of the facility and the amount of group processing time required by each test cell, lack of any requirement for fast real time response makes it possible to connect a large number.

10.6 Software Organisation

The actual details of the software will depend on the application of the system but it is possible to make a few points about the way in which the programming should be organised.

The software for the Level II processors should be fixed. Its actual operation would be controlled by using a configuration table approach. The fixed memory would also include a 'default' table, which is to be used unless the higher levels of the system define a new table for temporary usage whilst stored in read/write memory. Thus for the Level II A system the software would consist of generalized control algorithms with a few specialised routines such as the ramp generator. The configuration table would consist of two different sets of data, static and dynamic. The static table is used to provide the basic system configuration. It would consist of an entry defining the number of control loops and a number of sub-tables, one for each of the control loops. These sub-tables

would in turn contain information such as which data parameter is to be controlled by that loop, the output channel to the relevant actuator, the frequency of control updating, flags to include special control functions, etc. Associated with each loop there would also be a dynamic sub-table containing such information as the current set-point, the various loop gains and a flag to define whether or not ramp sequencing is to occur.

The input section, Level II B, would operate similarly except that its data tables would contain information of the frequency of logging, calibration and conversion factors, digital filtering constants, failsafe program activation flags and the relevant alarm and shutdown limits. The operation of the software in both processors would be on a task interleaving basis rather than by multiprogramming.

The software operations of the Level I system are somewhat different for the only fixed program is a simple bootstrap to allow the operator to request service by the group minicomputer. This then outputs a complete core-image package for the test to be run in the Level I processor. The microprocessor then performs the whole test by itself before finally transmitting the results back to the group minicomputer.

In its foreground mode the minicomputer would need to be able to search its library files and send the appropriate package to the releyant test cell. It must also be able to receive test results and haye printing and graphical plotting routines. It would also be used to generate the core-image packages for the microprocessors. To assist in doing this it should support a comprehensive library of

subroutines which allow a programmer to define a test sequence in simple terms by listing a series of calls to subroutines. These should be written to provide an image of the test cell as a single unit with all the division into different processing responsibilities between the different hardware and software sections of the test cell automation equipment being performed by the computer. The background capabilities should also include simulation activities to assist in the de-bugging of the new test procedures.

10.7 Summary

This chapter uses the information gleaned from the work on various aspects of engine test automation described in earlier chapters to define the requirements for an up-to-date system.

The results obtained from the use of already existing systems provide the basis for analysing the way in which the various tasks interact to the detriment of the system. As a result, a structure of tasks grouped into various categories which are not harmfully interactive has been derived. By using the low cost microprocessor as a prime element in the system, it is possible to show how an automated engine test system could be organised so as to provide a high degree of flexibility combined with the ability to perform the whole spectrum of testing methods without producing programming difficulties.

Chapter 11

CONCLUSIONS

11.1 Conclusions

The first generation of automated engine test systems has primarily been based on the use of steady-state conditions. This situation resulted mainly from the fact that the testing methods used were akin to those developed for manually controlled operations. Even so, the type of automation philosophy that was mostly widely used, that of several test beds with local analog controllers, supervised by a single minicomputer, gave rise to certain problems in practical operation despite the apparent simplicity of the system.

The concept of dynamic testing has its roots in the use of modern control equipment to perform functions that could not possibly be operated on a manual basis. There are two distinctive ways in which dynamic testing can provide an improvement on conventional steady-state methods. Firstly the use of dynamic ramps can speed up testing, hence reducing the cost of tests compared to the steady-state version. At the same time the use of a series of dynamic ramps to form the basis of a cycle can give a much more realistic appraisal of performance of an engine in typical usage, especially in view of the growing prevalence of urban driving conditions. This latter form of testing is of particular importance in relation to the measurement of exhaust emission levels for legal controls.

Despite its importance, the amount of work done on dynamic testing is still very limited. Most investigations have tended to be restricted to specific cases of particular tests and engine/

dynamometer combinations and hence their findings are not necessarily universally applicable. The program of investigation described in this thesis is intended to provide the basic information necessary to enable the definition of the next generation of engine test systems. Particularly attention was paid to the dynamic testing aspects to consider the feasibility of using such methods and the problems involved therein.

The findings of the work were:-

1. The use of a speed ramp provides an alternative to the conventional steady-state full-load power curve resulting in a reduction of test duration. Provided the acceleration rates are not too high, most data parameters show good correlation between steady-state and dynamic results. The hardest to correlate are the diesel smoke emission readings, but it is suggested that the dynamic results may give a truer indication of typical engine behaviour than their steady-state counterparts. A range of engines was used to allow generalisation of the findings.

2. The U.S.E.P.A. Smoke Cycle for heavy-duty diesel engines was used to investigate the automation requirements for emission testing. It was shown that it is possible to perform the required cycle using only very simple computer supervision provided that the local analog controller settings have been optimised. Alternatively the minicomputer can be used to provide more complex outer digital control loop abilities to decrease the demands put upon the perform-

ance of the local controllers. One possible method of doing this is by learning from the results of previous attempts to run the cycle. The technique of doing this in a fully automated way still needs further development, but, with some human input, the outer loop system can already provide the required behaviour.

3. In developing new testing techniques and procedures, the use of simulation methods can reduce cost and increase safety. For evaluation and de-bugging of dynamic testing systems, the simulation should operate in real time which, at present, means the use of analog rather then digital simulation. It is also important not to over-complicate the simulation but to use the simplest solution that will give adequate performance in the particular application.

Digital simulation on the other hand may be used for analytical simulation where the whole system, rather than just a part of it, is being simulated and there is no requirement for the time scaling to be exact. An extension of this work is the optimization of the handling system networks found in large engine production test facilities. A digital simulation technique that allows such networks to be studied has been developed and the mathematical representation of the various elements of the system could also form the basis of the control for an optimized facility.

4. The method of implementing other elements of engine testing practice, including conventional steady-state methods and diagnostic testing has been demonstrated. On the basis of this and the

findings above it is recommended that each test-bed must have its own supervisory control element and that the microprocessor is the obvious candidate for this task. Furthermore it would also be possible to replace nearly all the elements of the automation system by digital processing systems to provide a very flexibly structured philosophy based on software rather than hardware definition of system behaviour.

11.2 Suggestions for Future Work

There are a number of areas in which further work could advance the overall concept of an integrated philosophy of engine test automation. The question of fuel consumption measurement during dynamic tests needs to be examined. With the aid of an instantaneous measuring system the type of tests using the dynamic power curve and steady-state version should be repeated to add a correlation of fuel consumption measurements to those of torque and smoke readings.

There is room for a more comprehensive examination of the use of outer digital loops in connection with local controllers (whether analog or digital in nature) to provide a fully automated self-optimising control system. It is suggested that a useful approach which could be used is to develop a system that makes allowance for interaction between different control parameters. For instance, in the U.S.E.P.A. Smoke Cycle, the outer digital loop for speed control should also have some form of forward looking algorithm to predict the effect on speed control due to changes in throttle position.

In developing the control of an optimised handling system, the actual method of information transfer between the track and the con-

trolling facility should be examined. The possibility of using a distributed processing network based on microprocessors should be considered in order that single line serial data communication could be used to reduce the complexity, and hence the cost, of the system cabling, a factor which tends to be a major drawback of large scale centrally automated handling systems.

It is felt that the outline of a microprocessor based engine test automation system provides the basic blueprint to allow such a system to be built. Obviously a certain amount of detailed hardware design work is still needed. The matter of the actual detailed software however is a candidate for a major investigation to verify the overall feasibility of the concept. It may well be that in light of such work, some of the basic structure of the system may need modification.

REFERENCES

1. "Council Directive of 2nd August 1972 concerning the approximation of laws of Member States relating to the measures to be taken against the emission of pollutants produced by the diesel engines of motor vehicles". Official Journal of the European Communities No. L/190, 20th August 1972.

2. Federal Register, USEPA, Nov. 1970, Vol. 35, No. 219.

3. Federal Register, USEPA, Oct. 1971, Vol. 36, No. 193, Pt. I.

4. "Regulation re. the Measuring of Smoke Density in Exhaust Gases from Diesel Engines", National Swedish Road Safety Board, April 1969, Regulation F19-1969.

5. AGGETT, G. and HECK, R. "Automation of Test Rigs using Process Computers", Proc. Second International Symposium on Automation of Engine and Emission Testing, Sept. 1973, Vol. 1.

6. AL-BERMANI, S.A. "Computer Control of Diesel Engines for Pollution Testing", Ph. D. Thesis, University of London, 1975.

7. AVL — "AVL System 700 Fuel Comsumption Measuring Equipment Operating Manual"

8. AVL — "AVL 412 Smoke Meter Operating Manual".

9. AWNY, M.M. — "Optimal Computer Control of Engine Test Rigs", Ph. D. Thesis, University Of London, 1975.

10. BALESTRINO, A., SCIAVICCO, L. and EISINBERG, A. — "A New Approach to Modelling of Internal Combustion Engines", Proc. Fourth International Symposium on Engine Testing Automation, Sept. 1975, Vol. 2.

11. BANKS, K.W. — "An Automation Strategy for a Large Engine Test Facility", Proc. Second International Symposium on Automation of Engine and Emission Testing, Sept. 1973, Vol. 2.

12. BARRON, D.W. — "Computer Operating Systems", Chapman and Hall Ltd., 1971, pp. 46-78.

13. BENDER, R. — "Process Control System for Data Acquisition and Control of Test Stands. in the Development Section of a Motor Vehicle Factory", Proc. International Symposium on Automation of Engine Testing, Sept. 1972, Vol. 2

14. BIALYNICKI, M., CICHY, M. and MAZUREK, S. "Microcomputer Based Data Acquisition and Control System for Internal Combustion Engine Test Cells Automation", Proc. Fourth International Symposium on Engine Testing Automation, Sept. 1975, Vol. 2.

15. BONAMICO, P. "Microcomputer Automation System for Engine Test Rooms", Proc. International Symposium on Automotive Technology and Automation, Sept. 1977, Vol. 2.

16. BONGIORNO, G. and BISCHOF, D. "Special Features and Experience in the Use of Process Computers for Test Rig Automation", Proc. Fourth International Symposium on Engine Testing Automation, Sept. 1975, Vol. 2.

17. DODD, A.E. and HOLUBECKI, Z. "The Effect of Atmospheric Conditions on the Performance of Turbocharged Diesel Engines", The Motor Industry Research Association, March 1970, Report No. 1970/8.

18. FARROW, I.K. "Vehicle Simulation on an Engine Dynamometer and its Application to Emission Development", Proc. Second International Symposium on Automation of Engine and Emission Testing, Sept. 1973, Vol. 2.

19. GABOLA, G.
MIGLIACCIO, M.
and
INNOCENTE, R.
"Transient Behaviour of Internal Combustion Engine Test Beds", Proc. Fourth International Symposium on Engine Testing Automation, Sept. 1975, Vol. 2.

20. GARDINER, L.H.,
HECKFORD, D.C.,
and LOWRES, E.J.
"Computer Control and Sequencing for Industrial Engine Test Beds", Proc. International Symposium on Automation of Engine Testing, Sept. 1972, Vol. 2.

21. GARDINER, L.H.
and
PRICE, W.E.
"Operating Experience with Computer Control of Engine Test Beds", Proc. International Symposium on Automation of Engine Testing, Sept. 1972, Vol. 2.

22. GARDINER, L.H.,
SOUTHGATE, E.H.
and REA, D.P.
"Recomputerization of a Multi-Engine Test Facility", Proc. International Symposium on Automotive Technology and Automation, Sept. 1977, Vol. 2.

23. GENERAL AUTOMATION INC.
"CAP-16 Assembler Reference Manual", General Automation Inc., 1970, Document 88A00146A.

24. GENERAL AUTOMATION INC.
"FORTRAN IV Reference Manual", General Automation Inc., 1970, Document 88A00148A.

25. GENERAL AUTOMATION INC.
"RTX-16 Real Time Executive User's Manual", General Automation Inc., 1970, Document 88A00143A.

26. GENERAL AUTOMATION INC. "SPC-16 Automation Computer Reference Manual", General Automation Inc., 1970, Document 88A00150A.

27. GOODFRIEND, H.J. "Automatic Engine Testing - Performance and Diagnostics", Proc. Second International Symposium on Automation of Engine and Emission Testing, Sept. 1973, Vol. 2.

28. GRAVESTOCK, R.E. "Real-Time Simulation of a Petrol Engine", Proc. International Symposium on Automation of Engine Testing, Sept. 1972, Vol. 1.

29. GRAVESTOCK, R.E. "Dynamic Control of Reciprocating Internal Combustion Engines", Ph.D. Thesis, University of London, 1975.

30. GRUNERT, P.A., SCHULTZ, R. and ACKERMANN, L. "Process Control Computer at IHC Engineering Centre - Engine Lab.", Proc. International Symposium on Automotive Technology and Automation, Sept. 1977, Vol. 2.

31. HALL, B.E. and HENDERSON, I. "An Electric Actuator for Throttle Control", Proc. International Symposium on Automation of Engine Testing, Sept. 1972, Vol. 1.

32. HENDERSON, J. "A Computerized Data Acquisition System", Proc. Second International Symposium on Automation of Engine and Emission Testing, Sept. 1973, Vol. 2.

33. HILLS, P.R. "An Outline of the Simon Simulation System", Proc. Brunel University Conference on Simulation, March 1966.

34. HISLOP, E.W. "The Dynamic Power Curve", Proc. Third International Symposium on Automation of Engine Testing, Sept. 1974, Vol. 2.

35. HISLOP, E.W., FARROW, I.K. and SOLIMAN, J.I. "The Development of a Dynamic Data Retrieval and Control System for Engine Testing", Proc. International Symposium on Automotive Technology and Automation, Sept. 1976, Vol. 2.

36. IRONSIDE, J.M. "Closed-Loop Torque and Speed Control for Cycle Driving on an Engine Test Bed", Proc. International Symposium on Automotive Technology and Automation, Sept. 1976, Vol. 1.

37. KIBBLE, D.J. "An Automated Multi-Cell Exhaust Emission Test System", Proc. Second International Symposium on Automation of Engine and Emission Testing, Sept. 1973, Vol. 2.

38. KLINGENBERG, H. and LIES, K.H. "Critical Survey of the Exhaust Emission Test Procedure for USA and Europe", Proc. Fourth International Symposium on Engine Testing Automation, Sept. 1975, Vol. 1

39. LOWRES, E.J. "An Engine Test Computer System Designed for Use by Operators and Plant Engineers", Proc. International Symposium on Automotive Technology and Automation, Sept. 1976, Vol. 2.

40. MAEDA, T., ONO, T. and OHIGASHI, S. "Automatic Testing Equipment and Diagnostic Apparatus of Engines", Proc. Third International Symposium on Automation of Engine Testing, Sept. 1974, Vol. 1.

41. MUELLER, H.R. "Future Trends in Handling and Storage Systems for Advanced Production Test Facilities", Proc. International Symposium on Automotive Technology and Automation, Sept. 1976, Vol. 2.

42. NAYLOR, T.H. "Computer Simulation Experiments with Models of Economic Systems", John Wiley and Sons, 1971, pp. 1-59.

43. NIVI, H. "Computerized Diagnosis of Engine Faults", Ph.D. Thesis, University of London, 1975.

44. PRUE, D.A. "Automatic Diesel Diagnostics for Trucks", Proc. International Symposium on Automotive Technology and Automation, Sept. 1976, Vol. 2.

45. RIDEAL, P.H.C. "Diagnosis of Vehicle Malfunctions by Digital Computer", Ph.D. Thesis, University of London, 1976.

46. SANDTORV, H., FISKAA, G. and GUNDERSEN, G. "Instrumentation and Methodology for Computer Based Testing and Condition Monitoring of Diesel Engines", Proc. Third International Symposium on Automation of Engine Testing, Sept. 1974, Vol. 1.

47. SCHULZ, K. "Automation of Continuous Engine Testing Under Variable Test Conditions", Proc. Second International Symposium on Automation of Engine and Emission Testing, Sept. 1973, Vol. 1.

48. SCHWEIMER, G.W. "Emission Testing by Road Drive Simulation on Engine Dynamometers", Proc. International Symposium on Automotive Technology and Automation, Sept. 1976, Vol. 2.

49. SIMONIS, N. "Short-Test Engine Evaluation System", Proc. Third International Symposium on Automation of Engine Testing, Sept. 1974, Vol. 1.

50. SOLIMAN, J.I. and HISLOP, E.W. "Production Testing of Petrol and Diesel Engines", Proc. Fourth International Symposium on Engine Testing Automation, Sept. 1975, Vol. 2.

51. SOLIMAN, J.I. and HISLOP, E.W. "Engine Test Automation- Where to Now?", Proc. International Symposium on Automotive Technology and Automation, Sept. 1976, Vol. 1.

52. SOLIMAN, J.I. and HISLOP, E.W. "Computer Diagnosis of Motor Vehicles", Proc. International Symposium on Automotive Technology and Automation, Sept. 1976, Vol. 2.

53. SOLIMAN, J.I. and HISLOP, E.W. "A Microprocessor Based Engine Test System", Proc. International Symposium on Automotive Technology and Automation, Sept. 1977, Vol. 2.

54. SOLIMAN, J.I. "Advances in Engine Testing", Proc. International Symposium on Automation of Engine Testing, Sept. 1972, Vol. 1.

55. SOLIMAN, J.I. "Dynamic Computer Control of Engine Test Rigs", Proc. Second International Symposium on Automation of Engine and Emission Testing, Sept. 1973, Vol. 1.

56. SOLIMAN, J.I. "Advances in Engine, Emission and Diagnostic Testing", Proc. Third International Symposium on Automation of Engine Testing, Sept. 1974, Vol. 2.

57. SOUCEK, B. "Minicomputers in Data Processing and Simulation", John Wiley & Sons, 1972, pp 320-335.

58. SPOONER, W.J. "Precision Interface Conveying for 'On-Line' Automated Production Operations", Proc. International Symposium on Automotive Technology and Automation, Sept. 1977, Vol. 2.

59. TAYLOR, J.G. "Adaptive Control of Engine Test Rigs", Ph.D. Thesis, University of London, 1976.

60. WAHBA, S.A. "Optimal Control of Petrol Engines for Pollution Testing", Ph.D. Thesis, University of London, 1975.

61. WALKER, P.E. "The Introduction of a Computer Controlled Diesel Engine Test Facility into an Established Engine Lab.", Proc. Fourth International Symposium on Engine Testing Automation, Sept. 1975, Vol. 1.

62. WOOLNER, A. "Ford Approach to Automation", Proc. International Symposium on Automation of Engine Testing, Sept. 1972, Vol.2.